UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

TRABAJO FIN DE GRADO

SUSTITUCIÓN SENSORIAL: SISTEMA DE AYUDA A INVIDENTES

Javier de Pedro Pascual
Tutor: Guillermo González de Rivera Peces

MAYO 2020

SUSTITUCIÓN SENSORIAL:
SISTEMA DE AYUDA A INVIDENTES

AUTOR: Javier de Pedro Pascual
TUTOR: Guillermo González de Rivera Peces

Hardware and Control Technology Laboratory HCTLab

Dpto. Tecnología Electrónica y de las Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Mayo de 2020

Resumen (castellano)

Este Trabajo Fin de Grado pretende continuar con el trabajo de Fin de Máster de José Antonio Torre, el cual diseñó un prototipo para el Dispositivo de Sustitución Sensorial desarrollado por investigadores del Laboratorio de Visión y Percepción de la Facultad de Psicología de la UAM.

El objetivo de ese dispositivo es el que, una vez desarrollado por completo, pueda ser utilizado por personas con discapacidad visual para evitar obstáculos con los que se vayan encontrando al moverse. Para ello, estaría dotado de un chaleco con varias filas y columnas de motores vibrotáctiles los cuales vibrarían con distintas intensidades en función de la dirección y la distancia a la que estuviese el obstáculo con el que la persona va a chocar.

Este dispositivo contiene varias etapas: la adquisición de datos del entorno mediante cámaras, el tratamiento de esos datos en una unidad de procesamiento y su traducción para que, finalmente, se entreguen a los chips que controlan los motores por medio del protocolo I2C.

Tanto este TFG como el TFM de José Antonio [16], se centran en las dos últimas etapas comentadas, ya que la adquisición y tratamiento de datos se mantienen tal y como estaban en el prototipo de los psicólogos.

Este TFG en concreto, se centrará en adaptar el software desarrollado en el TFM para que se puedan utilizar varias filas de motores. También, se llevará a cabo el montaje del diseño de PCB flexible que se desarrolló en el TFM, pero no se llegó a construir. Además, se desarrollará una aplicación que aúne la configuración y el envío de datos a las filas de motores en una única interfaz.

Abstract (English)

This Bachelor Thesis aims to continue with José Antonio Torre's TFM, who designed a prototype for the Sensory Substitution System developed by researchers from the Vision and Perception Laboratory of the UAM's Faculty of Psychology.

The goal of this device is that, once it is fully developed, it could be used by people that suffer from visual disability to avoid obstacles that they find moving around. To achieve this, it would be equipped with a vest that would integrate rows of vibrotactile motors which would vibrate with different intensities depending on the direction in which the obstacle would be placed.

This device contains several stages: the acquisition of data depending on the environment, the treatment of this data in the processing unit, and the translation and delivery of that traduced data to the chip that controls the behavior of the vest´s motors through I2C protocol.

Both this TFG and José Antonio's TFM [16], are focused on the two last stages, considering that acquisition and processing phases maintain as they were developed by the researchers of the Vision and Perception Laboratory.

This TFG, is going to focus on adapting the software that was developed for the TFM for it to be useful to control several rows of motors. It will be also accomplished the set-up of the flexible material PCB, which was designed in the TFM, but it was never built.

Besides, an application that is capable of handling both configuration and delivery of data to the rows of motor, is going to be developed.

Palabras clave (castellano)

Sistema de Sustitución Sensorial, Actuadores Vibrotáctiles, XBee, I2C, PCA9685, AtmelStudio, Visual Studio, maestro, esclavo, barra de desplazamiento, botón, PCB, C, C#.

Keywords (inglés)

Sensory Substitution System, Vibrotactile Actuators, XBee, I2C, PCA9685, AtmelStudio, Visual Studio, master, slave, trackbar, button, PCB, C, C#.

Agradecimientos

Quiero comenzar dando las gracias a mi tutor Guillermo por su gran ayuda y su paciencia en la realización de este trabajo, en la que no he dejado de aprender.

También, me gustaría dar las gracias a mi familia, en concreto a mis padres y mi hermana Sara, que han estado siempre apoyándome y aguantándome. Sin su empuje en los momentos más difíciles no habría sido posible finalizar estos estudios.

Por último, quiero agradecer a Magali y a mis amigos, que han sido claves apoyándome y haciéndome la vida más fácil y divertida. En especial a aquellos que empezaron siendo mis compañeros de clase y se han convertido en grandes amigos.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

1 Introducción

1.1 Motivación

En la década de los 60, comenzaron a desarrollarse los llamados Sistemas de Sustitución Sensorial, los cuales tienen por objetivo sustituir o ayudar a alguno de los cinco sentidos que poseen los seres humanos con otro de los sentidos. Por ejemplo, en el caso de las personas que padezcan discapacidad visual, que la información que sería percibida por los ojos de esa persona, de alguna manera consiga llegar a otro de sus sentidos que sí que esté en perfectas condiciones de recibir información. Esa especie de "traducción" o "acomodación" de esa información visual a otro tipo de información perceptible por el sujeto, es lo que se encarga de realizar el Sistema de Sustitución Sensorial.

Aunque este tipo de dispositivos están siendo utilizados para otro tipo de aplicaciones como pueden ser las militares y de aviación, su principal uso y el más interesante es el de tratar de paliar las discapacidades sensoriales que sufren millones de personas para conseguir así que su vida sea más fácil y puedan llegar a tener las mismas oportunidades que las demás personas.

En concreto, debido al contenido de este TFG, nos centraremos en la ceguera.

Según un estudio llevado a cabo por la Organización Mundial de la Salud, se estima que a nivel mundial más de mil millones de personas tienen alguna deficiencia visual, de las cuales doscientos millones tienen una deficiencia visual de moderada a grave y 36 millones son ciegas.
Este número de personas comparado con la variedad y cantidad de Sistemas de Sustitución Sensorial a los que se pueden acceder en el mercado o están actualmente utilizándose es enorme. [1] [2]

Este TFG parte de un proyecto de colaboración con investigadores del Laboratorio de Visión y Percepción de la Facultad de Psicología de la UAM. Este departamento ha desarrollado un dispositivo de sustitución sensorial, que envía a una persona con discapacidad visual información acerca de la posición y la distancia a la que están los objetos que le rodean a través de una matriz de actuadores colocados en un chaleco que iría en el pecho del sujeto.

1.2 Objetivos

El prototipo desarrollado actualmente, en un trabajo previo de la EPS, permite el control de una fila de motores vibradores, a través de un sistema de radio basado en el protocolo ZigBee.

Los principales objetivos de este Trabajo de Fin de Grado son los siguientes:

- Extensión de ese trabajo a una matriz de motores de forma que, por un lado, reciba la intensidad de vibración de cada uno de los motores de la matriz y por otro, actúe sobre cada uno de los motores con dicha información. Dicho de otra forma, adaptar

el software existente para que sea capaz de controlar más de una fila de motores, en concreto, de una a ocho filas.

- Realización de un diseño modular del prototipo formado por dos etapas: la primera formada por el microprocesador y el módulo de radiocomunicación ZigBee y un segundo sistema que se encarga de la activación de los actuadores.

- Desarrollo de una aplicación para probar el prototipo que aúne tanto la configuración de los datos como el envío de estos al sistema encargado de activar los motores del chaleco.

- Evaluación del funcionamiento del prototipo conseguido con estas nuevas introducciones.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- **Capítulo 1:** En él se lleva a cabo una pequeña introducción acerca de los Sistemas de Sustitución Sensorial, dándoles una perspectiva general y explicando cómo surgió este trabajo y a qué necesidades responde.

- **Capítulo 2:** Recoge el estado del arte de los Sistemas de Sustitución Sensorial, comenzando primero por una clasificación de estos y centrándose finalmente en los sistemas táctiles y sus prototipos que se han ido desarrollando sobre todo los últimos años en distintos proyectos repartidos a lo largo del planeta.

- **Capítulo 3:** Se centra en el diseño que se encontró a la hora de comenzar este trabajo. Se dan a conocer los distintos módulos que componen el prototipo, así como los dispositivos que componen cada módulo, los protocolos que utilizan y los diseños de Altium que se construirán.

- **Capítulo 4:** Reúne todo el desarrollo que se ha realizado en este TFG: desde la revisión de los componentes y la sustitución de los que ya no funcionen hasta la creación de la aplicación de Windows que absorbe la configuración y el envío de datos; pasando por la adaptación del software para su uso con una matriz de motores y el montaje de los PCBs de material flexible.

- **Capítulo 5:** En él se recogen todas las pruebas realizadas con las nuevas implementaciones y se comenta si los resultados son válidos y son los esperados. También puede considerarse este capítulo como una guía de uso del dispositivo, ya que se comentan detalladamente los pasos que se siguen para realizar dichas pruebas.

- **Capítulo 6:** Se comentan las conclusiones que se obtienen de la realización de este TFG y se proponen posibles trabajos de futuro en lo que a él respecta.

2 Estado del arte

2.1 Historia de la sustitución sensorial.

La sustitución sensorial es el uso de un sentido humano para conseguir la recepción de información que de forma normal es recibida por otro sentido. Aunque es una disciplina que tiene numerosas aplicaciones, la mayoría de ellas tienen que ver con la ayuda a personas con algún tipo de discapacidad.
Se trata de un campo de estudio que contiene temas de psicología, neurociencia y por supuesto, de ingeniería.

El Braille, que se creó en el año 1840, se puede considerar el invento más popular de la historia en cuanto al campo de la sustitución sensorial.

En el año 1969, el neurocientífico americano especializado en plasticidad cerebral Paul Bach-y-Rita [3] publicó por primera vez sobre sistemas de sustitución sensorial. Se le considera el pionero de esa disciplina, ya que desarrolló el Tactile Visual Sensory Substitution, el primer sistema de sustitución sensorial basado en una interfaz humano-máquina. Bach-y-Rita utilizó la plasticidad cerebral como medio para aprovechar el sentido del tacto con el fin de obtener información del entorno, de manera que esta pudiera ser usada como sustitución del sentido de la vista. Partiendo de esta base, creó el TVSS, el cual fomentaba el uso de la plasticidad cerebral en pacientes con ceguera congénita.
Esa invención sentó la base para que la sustitución sensorial haya sido una pieza clave en numerosas investigaciones y avances científicos, sobre todo en el campo de la neurología cognitiva y perceptiva.

2.2 Funcionamiento y clasificación de sistemas de sustitución sensorial.

Los sistemas de sustitución sensorial suelen estar dotados de sensores cuya función es recopilar datos del entorno. Dichos datos se envían a un sistema capaz de interpretarlos y de actuar en consecuencia poniendo en funcionamiento actuadores que sustituyen la acción de uno de los sentidos estimulando otro de ellos.

Por el momento, se han desarrollado tres tipos distintos de sistemas de sustitución sensorial:

- Sistemas de sustitución sensorial táctiles.
- Sistemas de sustitución sensorial auditivos.
- Implantes de sistemas nerviosos.

2.2.1 Sistemas auditivos.

Los sistemas auditivos de sustitución sensorial son aquellos que permiten que, información que nos llegaría a través de otro sentido, nos llegue mediante el oído.
Los más importantes son los sistemas de sustitución sensorial auditiva-visual.
Los sistemas de sustitución sensorial auditiva-visual se basan en la sustitución del sentido de la vista mediante el sentido auditivo.
Un buen ejemplo de este tipo de sistemas es el vOICe, (un software creado a principios de los `90 por el investigador holandés Peter Meijer), formado por una cámara que adquiere las imágenes ópticas y una interfaz que las convierte en imágenes sonoras, las cuales se transmiten a unos auriculares. Este sistema extrae los parámetros característicos de una imagen, tales como el tono y el brillo, y los convierte en parámetros propios del sonido, como son la frecuencia y la intensidad. [5]

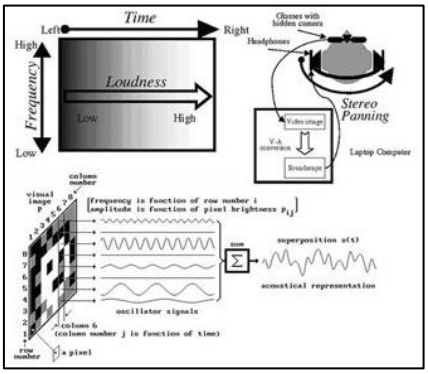

Figura 2-1: Esquemático resumen del sistema creado por Meijer.

2.2.2 Implantes de sistemas nerviosos.

Los implantes de sistemas nerviosos que más se han desarrollado son los implantes cerebrales, que están enmarcados dentro de un campo de investigación más amplio llamado interfaces cerebro-ordenador. Destaca sobre todo su uso en el campo de la visión, que, gracias al gran dominio científico que se ha alcanzado en el tema del sistema visual, se ha conseguido de forma exitosa que los implantes oculares se apliquen. En cuanto la audición, se utilizan implantes cocleares para estimular el nervio auditivo.

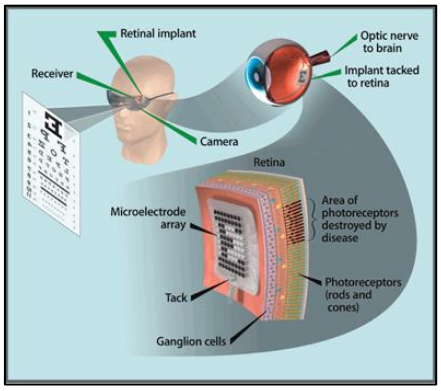

Figura 2-2: Esquemático resumen de sistema de implante nervioso.

2.2.3 Sistemas táctiles.

Vamos a centrarnos en los sistemas táctiles, ya que son en los que se basa este Trabajo de Fin de Grado.

Los sistemas táctiles de sustitución sensorial permiten percibir mediante el sentido del tacto información que normalmente llegaría a nosotros mediante otro sentido.

Trasladándonos a la base fisiológica de este tipo de sistemas, debemos conocer que nuestra piel posee distintos receptores táctiles. Por mencionar alguno, entre ellos están las terminaciones de Ruffini, las terminaciones nerviosas de Merkel y las terminaciones nerviosas libres.

Los receptores de la piel se diferencian normalmente dependiendo de la clase de estímulo que provoque que estos se activen.

Se pueden diferenciar dos tipos de estimuladores:

Por un lado, están los estimuladores electro-táctiles, que funcionan con estimulación eléctrica directa de la terminación nerviosa en la piel. Dependiendo del voltaje, corriente, forma de onda y muchos otros factores aplicados, la sensación de presión o quemadura sobre los receptores de la piel será mayor o menor.

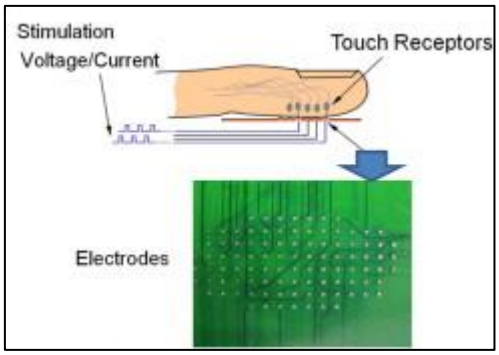

Figura 2-3: Esquemático resumen de estimulador electrotáctil.

Por el otro lado, se encuentran los estimuladores vibrotáctiles, que usan presión para activar los mecanorreceptores de la piel.

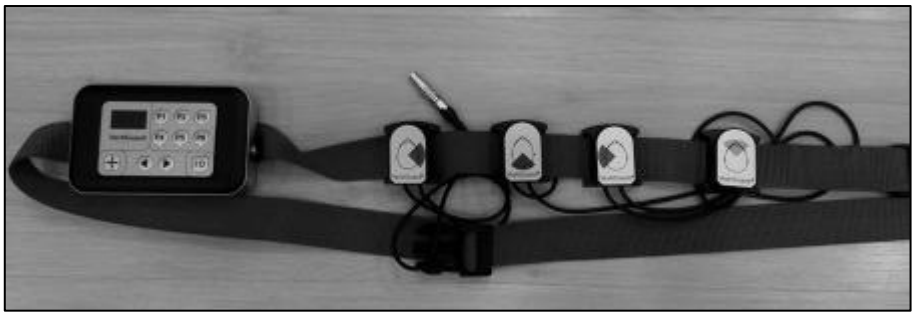

Figura 2-4: Ejemplo de estimulador vibrotáctil.

Hay varios tipos de sistemas de sustitución táctil, pero los más importantes son los sistemas táctil-visuales y los sistemas táctil-auditivos. Es en los primeros en los que se va a fijar el foco, ya que el dispositivo con el que se ha realizado en este TFG pertenece a ese grupo. En el siguiente punto, se va a hacer un repaso a los dispositivos de este tipo existentes hasta el día de hoy.

2.3 Dispositivos de sustitución sensorial táctil-visual.

Los dispositivos de sustitución sensorial táctil-visual están orientados a poder ser usados por personas con discapacidad visual, transmitiendo la información que el sujeto percibiría por el sentido de la vista para que sea adquirida por medio del sentido del tacto.
Como ya se ha comentado en el punto 2.1, el primer dispositivo de sustitución sensorial táctil-visual, y el que sentó las bases para los siguientes avances, fue el desarrollado por el neurocientífico Paul Bach-y-Rita en el año 1969.

2.3.1 TVSS: Tactile vision substitution system (Bach-y-Rita).

El TVSS hace posible convertir una imagen capturada con una videocámara en una "imagen táctil".

El sistema creado por Paul Bach-y-Rita [3] que se comentaba anteriormente estaba formado por una silla, cuyo respaldo era el encargado de "proyectar la imagen visual" gracias a sus 400 activadores, solenoides organizados en una disposición 20x20 (20 filas de 20 estimuladores cada fila.) Los estimuladores tenían una separación entre sí de 12 mm y 1 mm de diámetro, estaban construidos con teflón y vibraban contra la espalda del sujeto. Una vez sentada la persona que iba a usar el sistema, por medio de una cámara podía escanear todos los objetos y la disposición de la habitación y según la forma de estos objetos, los estimuladores de la espalda vibraban de un modo u otro.
Gracias a este sistema, se comprobó que las personas ciegas eran capaces de distinguir objetos y orientarse de forma relativamente rápida usando este prototipo. [5]

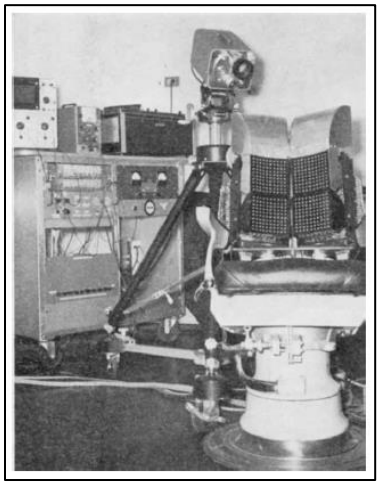

Figura 2-5: Fotografía del TVSS.

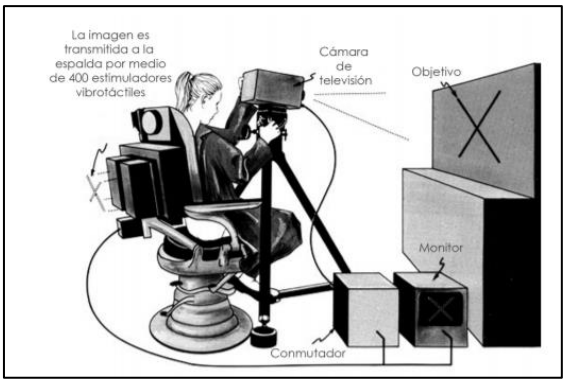

Figura 2-6: Esquema del funcionamiento del TVSS.

2.3.2 VideoTact.

El legado del TVSS fue tan importante que (eso sí, después de aplicarle muchas mejoras técnicas como el conseguir una mejor definición de imagen, mejorar la miniaturización, etc.) este prototipo sigue vigente y se encuentra disponible comercialmente bajo la marca de VideoTactTM (Unitech Research Inc). [4]

El VideoTact es un array electrotáctil multicanal construido en un sustrato conductor bastante fino llamado Kapton. Posee electrodos de titanio y está formado por dieciséis canales simultáneos que generan la forma de onda direccionados independientemente. Cada canal maneja 48 actuadores, por lo que en total hay 768 actuadores colocados en una disposición de 32 columnas x 24 filas.
Además, este aparato está preparado para ser controlado a través de una interfaz desde un PC al que a la vez se le suministraría la imagen que se quiere transmitir

2.3.3 TSAS (Tactile Situation Awareness System).

El TSAS [6] es un sistema que proporciona estímulos vibrotáctiles. Está formado por 22 actuadores neumáticos situados en un cinturón que se conecta a una batería.
Dicho prototipo comenzó a ser desarrollado como un sensor que ayudase a los pilotos de aviones del ejército estadounidense a remediar la desorientación que pueden sufrir en un momento dado. Además, este se ha ido mejorando y utilizando para otras aplicaciones, como, por ejemplo, permite indicar que un objeto se acerca al usuario (esta es la aplicación que más nos interesa para nuestro trabajo, debido a que esta función puede resultar útil para personas con discapacidad visual).

Figura 2-7: Fotografía de un piloto con el TSAS puesto.

2.3.4 Enactive Torch.

El dispositivo Enactive Torch es un sistema de sustitución sensorial que provee al sujeto que lo utiliza de una ayuda para la orientación en el entorno. [5]
Este sistema está compuesto por un sensor de ultrasonidos e infrarrojos y una unidad vibrotáctil. Fue desarrollado en la Universidad de Cincinnati, y permite que las personas ciegas puedan orientarse detectando los objetos de su alrededor. Según la dirección y la distancia a la que se encuentre el obstáculo que esta persona debería salvar para no chocar, el sensor efectúa una vibración de mayor o menor intensidad en el actuador vibrotáctil.
Por último, hay que comentar que el uso de este sistema es muy fácil y los investigadores que lo desarrollaron aseguran que con 10 minutos de entrenamiento ya se puede usar perfectamente consiguiendo así no tener que usar un bastón físico y evitando por lo tanto el contacto directo con los objetos.

Figura 2-8: Fotografías de un Enactive Torch.

2.3.5 Haptic Radar.

El radar háptico, [7] (la háptica es la ciencia que estudia el tacto), es un proyecto que fue desarrollado en la Universidad de Tokio y cuyo prototipo se presentó en 2006.
Este sistema se basa en una arquitectura de red de módulos iguales entre sí, que consisten en un sensor de infrarrojos y una unidad vibrotáctil. Además de la estimulación individual de cada unidad vibrotáctil, se posibilita también la estimulación de forma global gracias a la conexión existente entre los módulos. Así es como, un sistema formado por un número suficiente de módulos podría considerarse como una "piel extendida en el espacio". El sujeto es capaz de reaccionar de forma muy rápida a los posibles estímulos.
El primer prototipo usado en este proyecto es una banda para la cabeza formada por seis módulos dispuestos alrededor de esta. Cada módulo utiliza un sensor de proximidad de infrarrojos que cubre un ángulo de 30 º y una distancia de 80 cm. Si uno de los sensores localiza un objeto, envía un estímulo a su unidad vibrotáctil correspondiente.

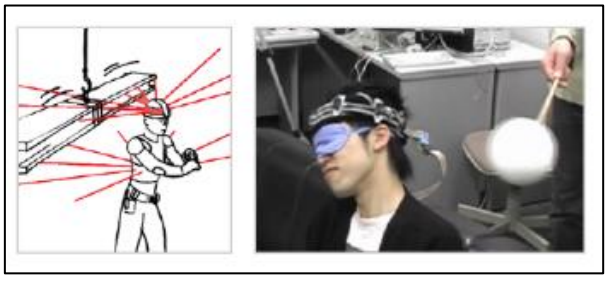

Figura 2-9: Fotografías del radar háptico.

2.3.6 Aural Antennae.

La Aural Antennae [5] (Antena Auditiva), es un sistema basado en la idea del Haptic Radar comentado en el anterior punto. La diferencia es que en este proyecto en vez de usarse rayos infrarrojos como en el caso del Haptic Radar, se utiliza un micrófono como sensor captador de la información de entrada. La segunda parte del prototipo es muy similar a la del Haptic Radar: una unidad vibrotáctl que produce estímulos vibratorios sobre la piel.

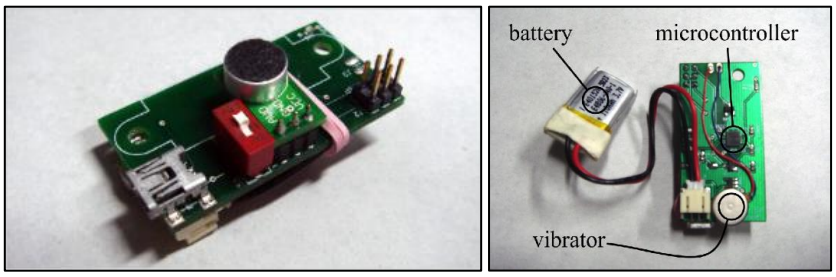

Figura 2-10: Fotografías del sistema Aural Antennae.

2.3.7 Cinturón FeelSpace.

El sistema de sustitución sensorial FeelSpace [5] ha sido desarrollado intentando imitar el "poder sobrenatural" que poseen algunos animales en cuanto a sentido de la orientación. Este dispositivo sirve esencialmente para poder guiar al sujeto que lo utiliza a llegar a un destino, es decir, es un sistema de navegación.

Gracias a un sistema de brújulas, el sujeto que lo usa puede percibir estímulos relacionados con la orientación y con estímulos táctiles.

El sistema de brújulas está formado por distintos componentes como acelerómetros y sensores magnéticos, y es el encargado de entregar la información. La posición del Polo Norte se calcula continuamente y se transmite a la unidad vibrotáctil, la cual está formada por 13 motores vibratorios.

Figura 2-11: Sistema FeelSpace.

El último dispositivo creado con esta tecnología de FeelSpace tiene por nombre comercial naviBelt [8], e incorpora una función que puede resultar bastante interesante para las personas con discapacidad visual:

Por medio de Bluetooth, se puede conectar el cinturón a un dispositivo móvil Android o IPhone, y se puede introducir mediante comandos de voz el sitio al que nos querríamos desplazar y ya simplemente notaríamos hacia dónde habría que ir gracias al cinturón.

2.3.8 Guante CyberTouch.

El sistema del guante CyberTouch [9] se creó en la empresa CyberGlove Systems, que se fundó en el año 1990 con el objetivo de desarrollar tecnologías hardware y software que permitiesen a los usuarios el interactuar con un ordenador usando el sentido del tacto.
CyberTouch consiste en seis pequeños actuadores vibrotáctiles, cada uno situado en un dedo y el sexto en la palma de la mano. Cada actuador puede ser programado individualmente para generar la respuesta que se desee y tienen una resolución menor de 1 º.
Este sistema permite sentir objetos 3D generados por ordenador.

Figura 2-12: Guante CyberTouch.

2.3.9 Prototipo de la Universidad de Guelph.

El profesor John S. Zelek, desarrolló un prototipo interesante sobre todo por ser bastante económico y muy simple a la par que funcional. [10]
Este sistema está compuesto por módulos portátiles, de baja potencia e intercambiables en cualquier momento si uno se averiase.
El sistema se fundamenta en la incorporación de la información de profundidad adquirida gracias al uso de un tipo de cámaras, y la transformación de dicha información de profundidad en datos táctiles que sean de ayuda para personas ciegas mientras realizan un recorrido.
Como podemos observar en la fotografía, es muy similar a otros prototipos que hemos comentado anteriormente: está formado por un guante con cinco actuadores vibrotáctiles, dos cámaras y un procesador. Cada dedo corresponde a una dirección en el espacio. A cada componente vibratorio, se le asigna una determinada sección vertical por medio de un algoritmo. Si se halla un píxel en una determinada zona que corresponda a un actuador, se hará vibrar a ese actuador, informando así a la persona invidente de que existe un obstáculo cercano en la dirección del actuador que vibre.

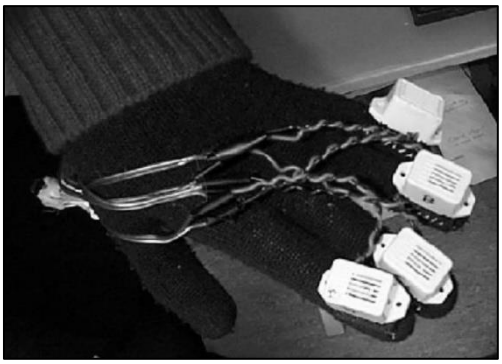

Figura 2-13: Proyecto de la Universidad de Guelph.

2.3.10 Proyecto de la Universidad de Wollongong.

El sistema de visión electro-neuronal (ENVS), [11] fue desarrollado por Simon Meers y Koren Ward, investigadores australianos de la Universidad de Wollongong. En su sistema, se busca tener la capacidad de eludir obstáculos y se implementa la navegación en entornos al aire libre utilizando sensores visuales y tecnología GPS. Este sistema posee un auricular, dos cámaras estéreo, una brújula digital, un procesador de GPS, una base de datos con localizaciones, una unidad de control y unos guantes con actuadores vibrotáctiles.

Figura 2-14: Prototipo del ENVS.

3 Diseño

El prototipo a partir del cual se ha trabajado fue desarrollado por investigadores del Laboratorio de Visión y Percepción de la Facultad de Psicología de la Universidad Autónoma de Madrid. Basándose en el TVSS de Bach-y-Rita comentado anteriormente en el Estado del Arte, desarrollaron el dispositivo de sustitución sensorial TSIGHT (Tactile-Sight) [12]. El TSIGHT se diferencia del TVSS en que, mientras el TVSS variaba la vibración de los actuadores dependiendo de la luz captada por una cámara, el TSIGHT consigue que los actuadores vibren dependiendo de la distancia a la que se encuentren los objetos hacia los que la persona con discapacidad visual se acerque y en qué dirección se encuentren.

Después, el alumno del Máster Universitario en Ingeniería de Telecomunicación José Antonio Torre Albarsanz, desarrolló como Trabajo de Fin de Máster un nuevo prototipo manteniendo tal como estaba la parte de la adquisición y procesamiento de las imágenes recibidas por las cámaras. [16]

A continuación, se comentarán ambos prototipos, a partir de los cuales se ha realizado este Trabajo de Fin de Grado.

3.1 Primer prototipo.

El prototipo TSIGHT se basa en conseguir adquirir información del entorno a través de una cámara Kinect (de XBOX). Esa información adquirida se pasa por una unidad de procesamiento que se encuentra en un ordenador. Ese mismo ordenador se encarga de enviar la información a los actuadores por medio del protocolo ZigBee. Para esa comunicación, se intercala un microprocesador que actúa como dispositivo maestro de una comunicación I2C en la que los esclavos son los actuadores.

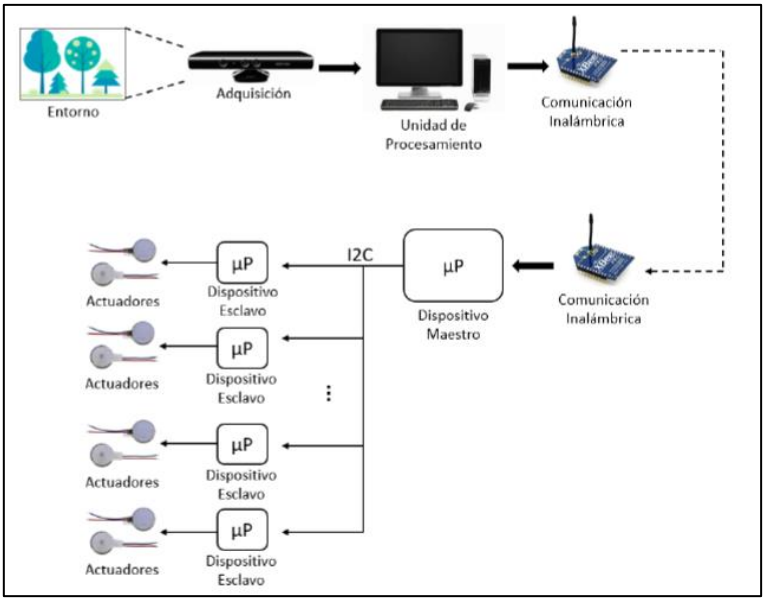

Figura 3-1: Esquema del TSIGHT.

Lo cierto es que este prototipo era funcional, pero como se comenta en [16], tenía unas dimensiones muy poco manejables y no acordes al uso que se le quería dar al sistema.
Por este motivo, se desarrolló un nuevo prototipo, con el objetivo de conseguir unas dimensiones más acordes a su objetivo.

3.2 Segundo prototipo.

El segundo prototipo, (el desarrollado en [16]), es a partir del cual se ha trabajado en este TFG. A continuación, se van a detallar los elementos que lo componen.

- La etapa en la que se adquieren y se procesan las imágenes recibidas por las cámaras se mantiene tal cual estaba en el primer prototipo.
- Dos dispositivos de comunicación inalámbrica que funcionan con el protocolo ZigBee. El primero se comunica en serie con el ordenador, es decir, la unidad de procesamiento de los datos recibidos. Este primer dispositivo envía por radio los datos deseados al segundo dispositivo, el cual está conectado en serie con un microprocesador mediante un módulo UART.
- Un microprocesador que funciona como maestro en un bus I2C encargado de manejar los actuadores (esclavos.)
- Un Driver I2C compuesto por las filas de motores del chaleco (actuadores), que obedecen a las instrucciones que les envía el maestro.

Seguidamente, se analizará cada módulo por separado y con mayor detalle, incluyendo los modelos de cada dispositivo que se utilizan.

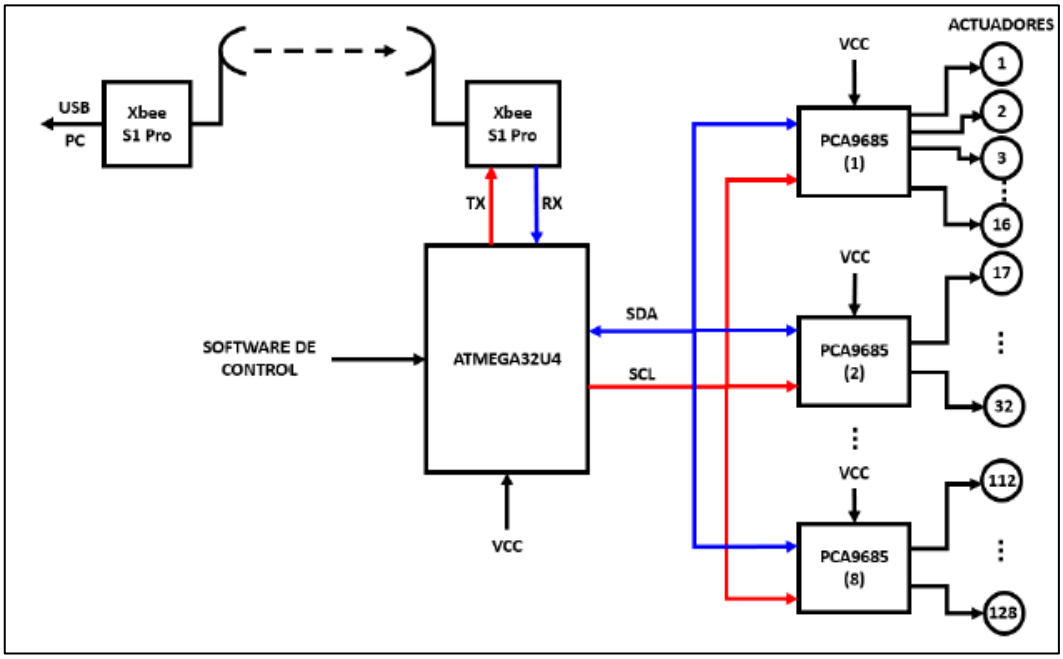

Figura 3-2: Diagrama del prototipo 2.

3.2.1 Módulos de radiofrecuencia.

Para la primera etapa del esquema, se eligieron dispositivos XBee de Digi International. En concreto, dos del modelo Xbee S1 Pro [13]. Estos dos dispositivos, se conectaban a través del programa de DIGI XCTU, software con el que se pueden configurar parámetros de la comunicación de radio. Si bien en este TFG al principio se utiliza XCTU para configurar ambos módulos de radio como en el TFM, uno de los objetivos como veremos posteriormente es la integración de todo lo necesario para realizar el envío de datos a los esclavos en una sola interfaz.

Figura 3-3: Módulo de radio XBee Pro S1.

3.2.2 Microprocesador.

El microprocesador, como se comentaba anteriormente, hace las veces de maestro en la red I2C y a su vez posee una conexión en serie con el receptor de radio. El microprocesador elegido fue el chip ATMEGA32U4 [14], un chip de la Atmel que se programa gracias a la plataforma Teensy 2.0 mediante el software Atmel Studio en lenguaje C. Usando ese programa, se pueden generar los ficheros .hex que se bajan a la placa teensy (desde el ordenador mediante mini-USB, utilizando una interfaz de teensy), y configuran el microprocesador para que este realice sus funciones correctamente. Para llevar a cabo esa configuración, se utilizan varias librerías, como la i2c.h, la Xbee.h o la PCA9685.h.

En el TFG, como se verá más adelante, se han tenido que realizar algunos cambios en este software que se desarrolló en [16] para adaptar el prototipo al uso de varias filas de esclavos (otro de los objetivos del TFG.)

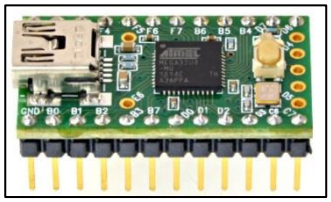

Figura 3-6: Teensy 2.0.

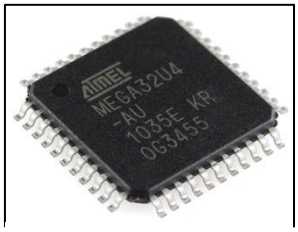

Figura 3-5: Chip ATMEGA32U4

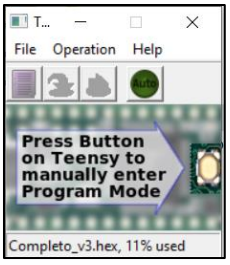

Figura 3-4: Interfaz de Teensy.

Para conectar los dispositivos nombrados anteriormente, se diseñó un PCB en el que se encajan el receptor XBee, la Teensy 2.0. y el chip ATMEGA. Además, para regular su

funcionamiento, se añaden nuevos elementos a ese PCB como un conector de alimentación, un regulador de tensión, seis condensadores y un conector I2C.

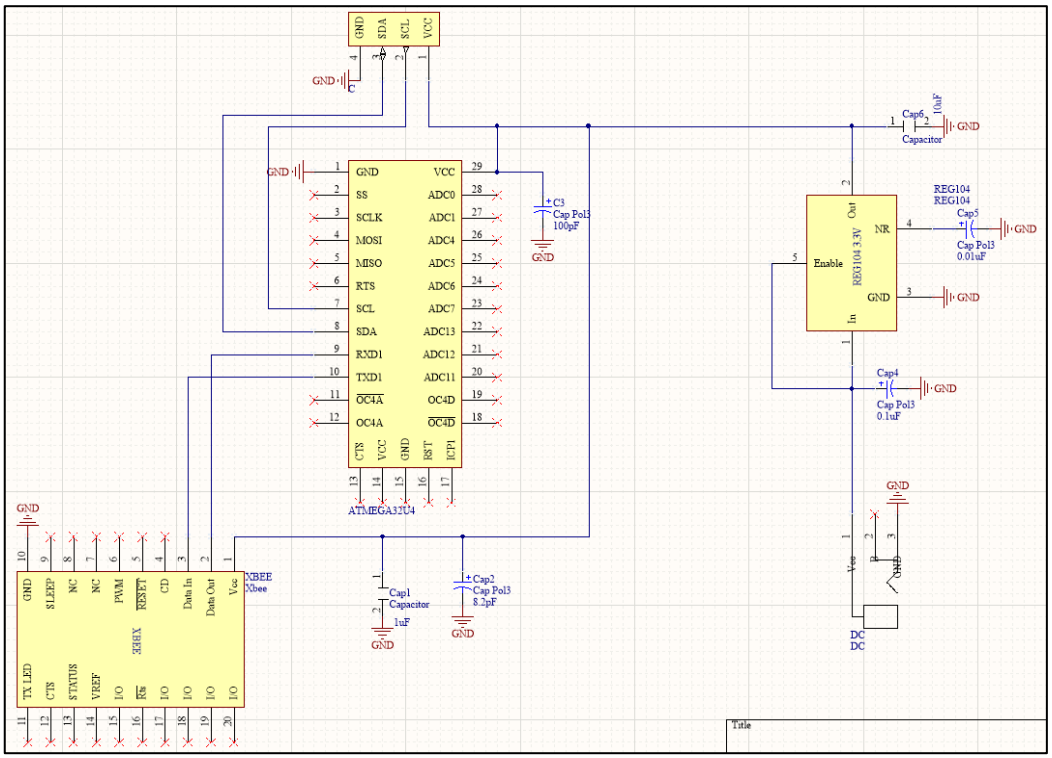

Figura 3-7: Esquemático del PCB en Altium.

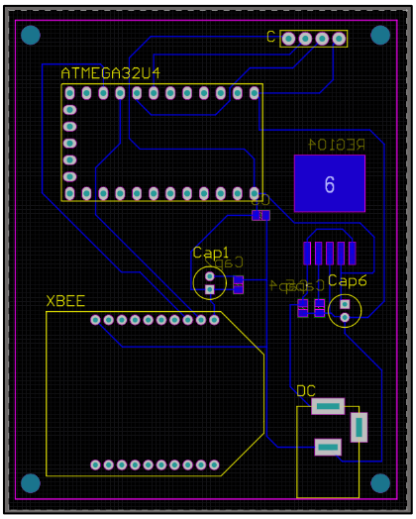

Figura 3-9: Huellas del PCB

Figura 3-8: PCB ya construido

3.2.3 Driver I2C.

Para la última etapa se utilizan chips PCA9658 de NXP [15]. Se utilizan tantos chips como filas de 16 actuadores se quieran construir. Este microchip maneja dieciséis canales controlados por un bus I2C.
El bus I2C está formado por cuatro vías:

- SDA (Serial Data): Pista que comunica al maestro y al esclavo para el envío de datos entre sí.
- SCL (Serial Clock): Pista por donde circula la señal de reloj (comunicación síncrona).
- VCC: Vía de alimentación.
- GND (masa): Común entre todos los dispositivos conectados al bus.

Como se puede ver en la figura de abajo, este chip está preparado para controlar la intensidad con la que lucen los leds conectados a las distintas salidas y en el diseño de Altium de [16] se permite la configuración de distintas direcciones I2C según se suelden los pines A0, A1 y A2.

Las salidas de los pines disponen de un control PWM, que posibilita que los actuadores vibren con distintas intensidades según los datos recibidos, o en este caso en el que testeamos con leds, que los leds brillen más o menos.

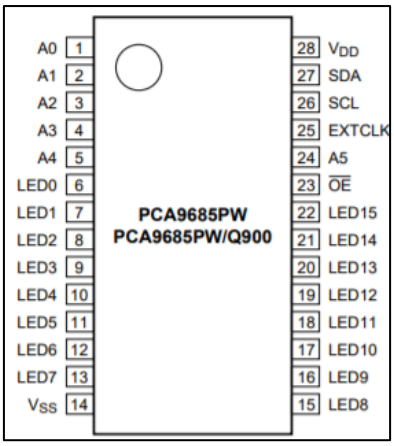

Figura 3-10: Pines PCA9685.

Para esta última etapa también se diseñó un PCB, primero de material normal que fue el que se utilizó en [16] para realizar pruebas, y en última instancia uno que no se llegó a construir en aquel momento, hecho de material flexible debido a que iría colocado en el chaleco, en el que va integrada la PCA9685, así como los leds con sus respectivas resistencias de protección, pads que según sean soldados nos permiten asignar una u otra dirección al PCB, resistencias y condensadores.

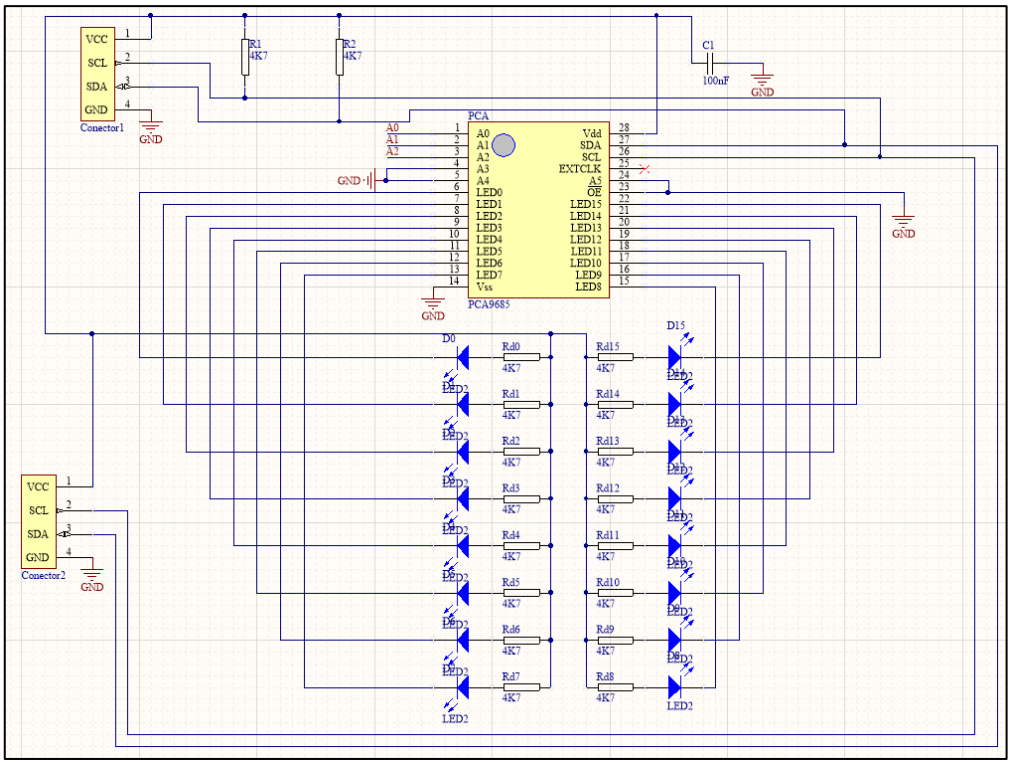

Figura 3-11: Esquema PCB flexible.

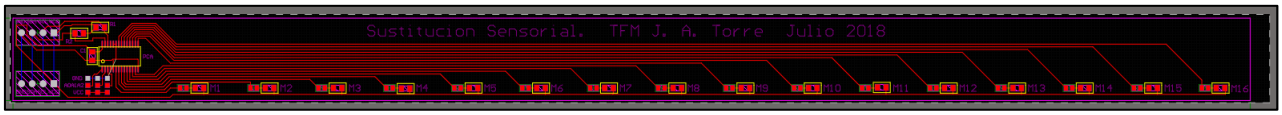

Figura 3-12: Diseño PCB flexible.

Como se van a utilizar varios PCBs flexibles (en este TFG se harán pruebas con tres, pero el diseño permite hasta ocho), es importante comentar que estas filas estarán unidas entre sí con un cable o soldadas juntando cada vía de sus conectores I2C (cada PCB dispone de dos conectores).

4 Desarrollo

El segundo prototipo descrito en el apartado 3.2. es el punto de partida para la realización de este TFG.

Para dar una visión general de las líneas de trabajo que se van a seguir y poder encuadrarlas en cada etapa del prototipo, se han numerado en la Figura 4-1 y se van a describir de forma breve inmediatamente debajo.

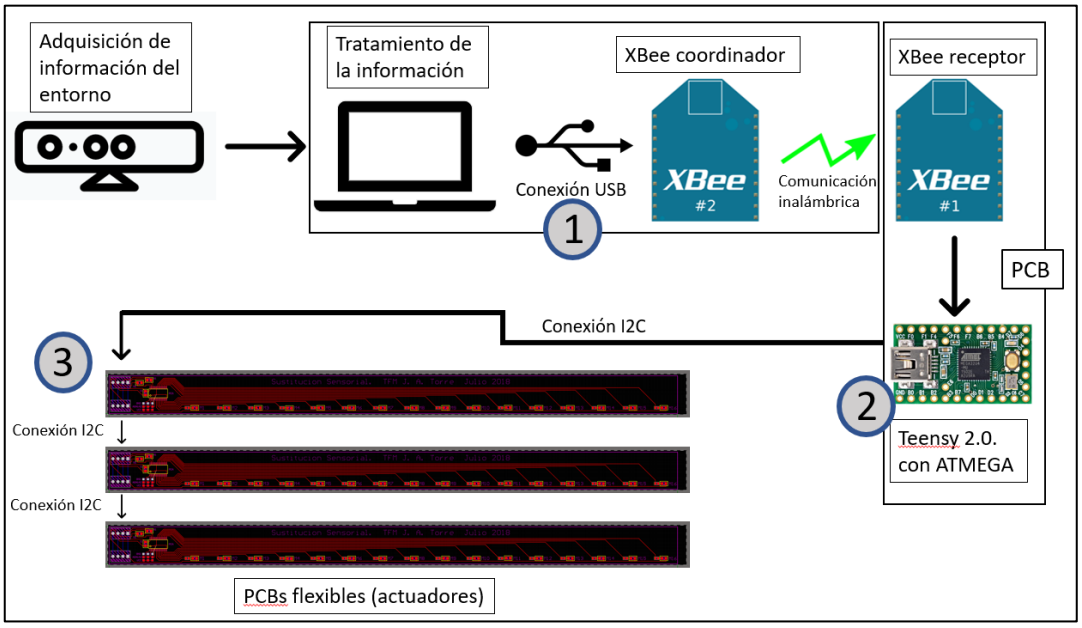

Figura 4-1: Esquema general con numeración.

1. Desarrollo de una aplicación de prueba que permita elegir la configuración de los actuadores y enviarla primero desde el PC hasta el dispositivo XBee coordinador mediante el puerto serie y seguidamente del XBee coordinador al receptor por radio.
2. Programación de la placa Teensy para que se puedan manejar varios PCBs flexibles por medio de la conexión I2C.
3. Montaje de los PCBs asignándoles distintas direcciones I2C.

4.1 Comprobación de dispositivos.

Inicialmente, se tuvo que comprobar que todos los dispositivos utilizados funcionasen de forma correcta. Uno de los dos dispositivos XBee Pro S1, en concreto el que se utilizaba como receptor, no funcionaba. Gracias a la escalabilidad de los distintos módulos, se pudieron cambiar por otro modelo, en este caso de una versión anterior, el XBee S1.
A continuación, se detallarán los pasos de configuración que se han tenido que seguir en el programa XCTU para poner los dos dispositivos XBee S1 en comunicación.

Para empezar, se conecta el dispositivo que se va a usar como emisor. Se escoge un canal, (en este caso C), y se fija un valor para el campo PANID que se copiará en el otro dispositivo.

Después, en el campo Coordinator Enable (CE), se selecciona la opción Coordinator [1], dándole así las funciones de coordinador de la comunicación.

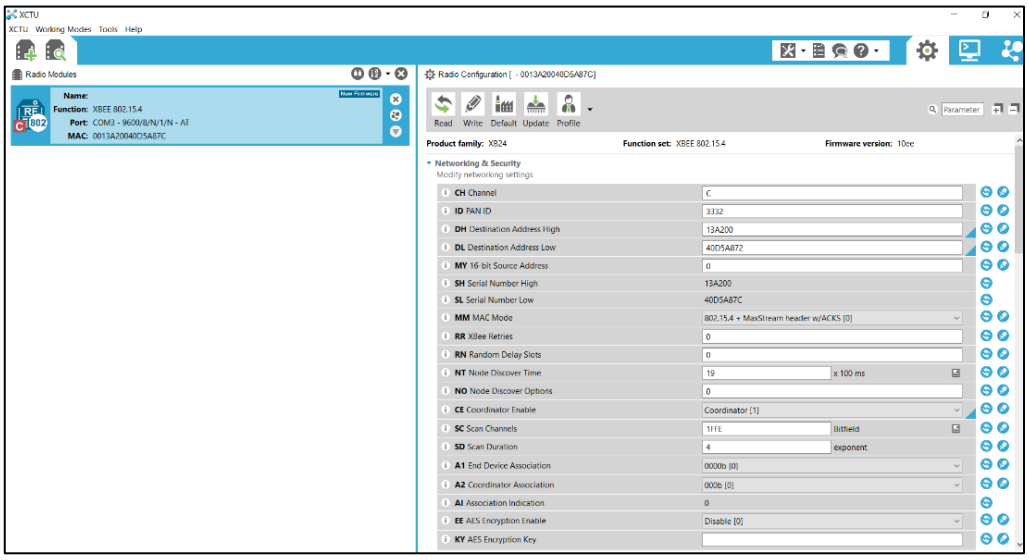

Figura 4-2: Configuración dispositivo coordinador en XCTU.

Una vez configurado el coordinador, se conecta el segundo dispositivo. A este, se le fija el mismo canal y el mismo PANID que se le ha asignado al coordinador y se le activa la opción de Channel Verification. En este caso, el campo Coordinator Enable se mantiene desactivado.
Finalizada la configuración de los dispositivos de radio, estos ya están preparados para que la conexión inalámbrica funcione correctamente.

4.2 Montaje de los PCBs flexibles.

A partir del diseño de Altium de [16], se lleva a cabo la construcción de los PCBs de material flexible.
Se sueldan los componentes descritos en el esquema de las figuras 3-11 y 3-12, (condensadores, resistencias, chip PCA y conector I2C.)
Para que cada PCB tenga una dirección I2C distinta y sean controlados por el maestro, el PCB dispone de tres pads, conectados a los pines de la PCA A0, A1 y A2, cada uno de los cuales se puede soldar a VCC o a GND, lo que permite definir ocho direcciones distintas.

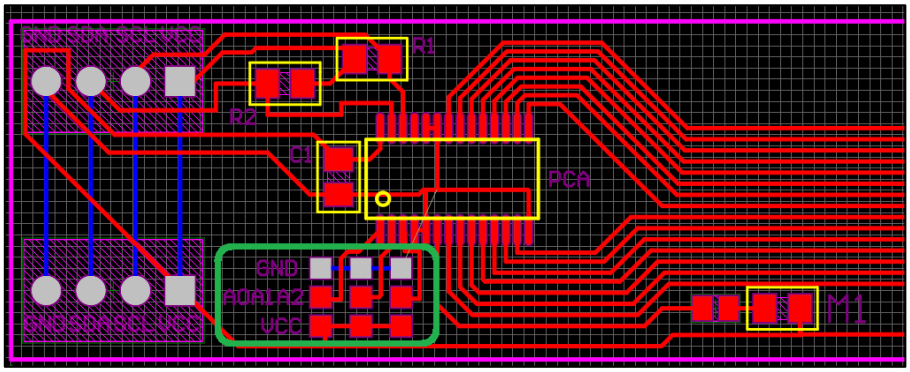

Figura 4-3: Pads encargados de generar direcciones recuadrados en color verde.

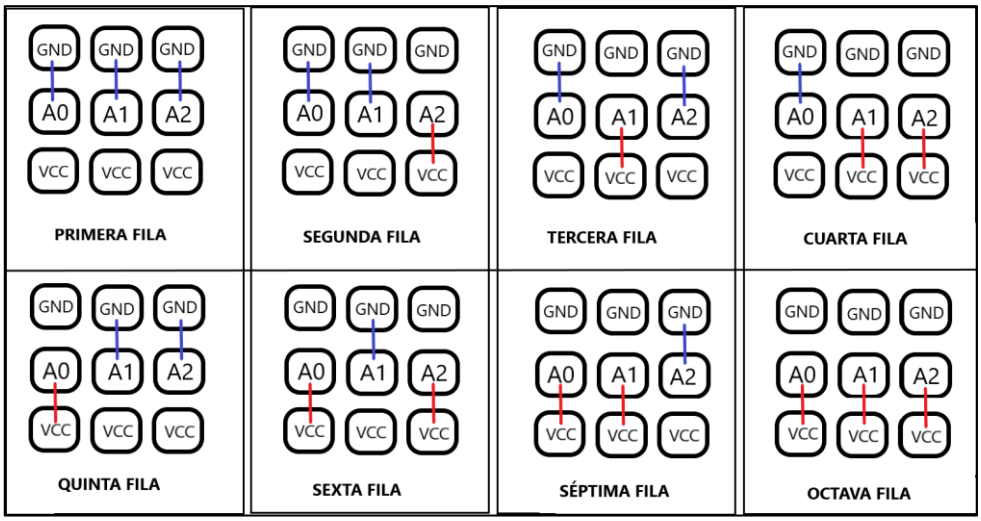

Figura 4-4: Asignación de direcciones.

Para este TFG se han construido las tres primeras filas:

	A0	A1	A2
Primera fila	0	0	0
Segunda fila	0	0	1
Tercera fila	0	1	0

Tabla 4-1: Direcciones programables.

Las direcciones de esclavo del chip PCA tienen el siguiente formato:

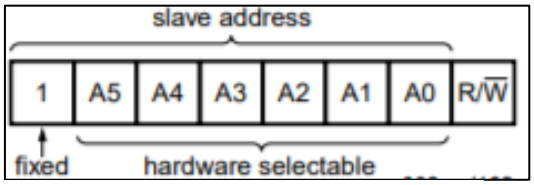

Figura 4-5: Formato de esclavos

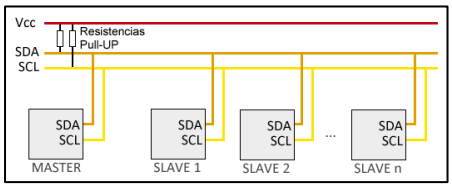

Figura 4-6: Esquema de un bus I2C

Los pines A5, A4 y A3 están fijados a 0 en todos los PCBs.
Por ello, la dirección del primer esclavo será 0x80 (1000 0000 en binario), la dirección del segundo esclavo será 0x82 (1000 0010) y la dirección del tercer esclavo será 0x84 (1000 0100), aumentando de dos en dos unidades en numeración hexadecimal en las sucesivas filas.

Hay que comentar que, como se puede comprobar en el código, (Anexo A), el parámetro que se le pasa a la función de enviar datos con la dirección del esclavo se debe desplazar un bit a la derecha, (por ejemplo, para la primera que es 0x80, se debe pasar 0x40).

Para finalizar el montaje de los PCBs, se elige qué tipos de leds utilizaremos y se calcula el valor que deben tener sus resistencias de protección. Hay que aclarar que, si bien en el

prototipo final estos leds serán sustituidos por actuadores vibrotáctiles (motores), para este TFG se usarán diodos led para realizar las pruebas.

Se eligen diodos led de encapsulado 0805. Estos leds, (fijándonos en la figura 4-7), tienen el ánodo en el lado hacia el que apunta el cuadrado verde.

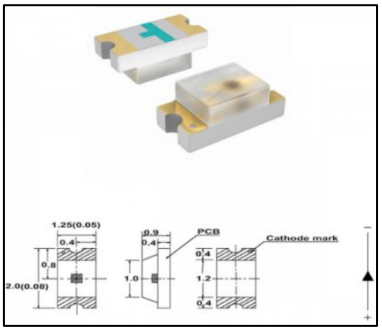

Figura 4-7: Diodos de encapsulado 0805.

Consultando su hoja de características para escoger las resistencias de protección correctas, se observa que, dado que tenemos una tensión Vcc = 5V y circula una corriente de diodo de 5 mA; y que además se puede ver que el diodo led va a tener una diferencia de tensión en sus bornes de 2 V, tendremos que escoger una resistencia del orden de 600 Ω.

V = Rprot * Idiodo; 3 V = Rprot * 5 mA; Rprot = (3 V)/(0.005 A) = 600 Ω

Se escoge el valor de resistencia de encapsulado 0805 más cercano a 600 Ω:
Un valor de resistencia de 560 Ω (código 561).

Figura 4-8: Resistencia de encapsulado 0805.

Una vez soldados todos los componentes de los PCBs, estos ya están listos para su uso.

4.3 Programación en AtmelStudio para múltiples esclavos.

Como se comentaba en el apartado 3.3.2, la plataforma Teensy (en la que se integra el chip ATMEGA que será el que funcione como maestro de la comunicación I2C), se configura a través de un fichero .hex que se genera al compilar el nuevo main.c que desarrollemos junto con las librerías que se crearon en [16] en el programa AtmelStudio.

El fichero main.c estaba preparado para comunicarse con una sola fila de actuadores, así que se ha modificado su código para poder controlar una matriz de varias filas de actuadores. En este caso, las pruebas las haremos con tres esclavos, pero en el código se han dejado

comentadas todas las variables y líneas de código necesarias, que deberán ser descomentadas en el caso de querer utilizar más de tres filas.

El código completo se puede consultar en el Anexo (Apartado A).
A continuación, se van a comentar las principales modificaciones que se han tenido que implementar.

- Como se decía en el apartado 4.2, las direcciones de los esclavos serán las siguientes:

```
//INICIALIZACIÓN DE DIRECCIONES DE LOS ESCLAVOS.
uint8_t addrMaster1 = 0x40;     //PRIMER ESCLAVO

uint8_t addrMaster2 = 0x41;     //SEGUNDO ESCLAVO

uint8_t addrMaster3 = 0x42;     //TERCER ESCLAVO
```

Figura 4-9: Definición de las direcciones de los esclavos.

- Una vez inicializado el driver I2C (llamando a las funciones de la librería PCA9685.h begin() y setPWMFreq() para cada una de las direcciones anteriores: addrMaster1, addrMaster2, …), se define un array de tamaño 48 (número de leds a configurar, es decir, 3 filas x 16 leds en cada fila). Además, se definen tres arrays de tamaño 16 (uno para cada fila.)
  Dentro de un bucle while, se utiliza el array de mayor tamaño para extraer los datos recibidos con la función de la librería Xbee.h uart_getchar(), y se asocia cada uno de los arrays de menor tamaño a su fila correspondiente. Es decir, usando bucles for, se reparten los bits que se han guardado en el array de tamaño 48 en los arrays de 16 bits que correspondan a cada tramo de bits.

- Como ya se tienen guardados los datos que corresponden a cada esclavo en sus respectivos arrays, con un bucle for se va asignando a cada led de cada esclavo el valor de PWM que le corresponde por medio de la función setPWM de la librería PCA9685.h.

Cuando ya se termina de desarrollar el código del main.c y habiendo incluido todas las librerías necesarias, se compila el proyecto pulsando en Build Solution y se genera el fichero .hex que se baja a la plataforma teensy para configurarla, tal y como se verá en el apartado de pruebas.

4.4 Desarrollo de aplicación de configuración.

Es importante señalar que esta aplicación se va a desarrollar para hacer pruebas con el prototipo, pero en la versión definitiva del sistema de sustitución sensorial, la aplicación que se usará para enviar a los motores el valor de PWM (cualquier valor entre 0 y 100 con el que se tiene que excitar cada motor), será la que se utiliza para procesar los datos que se adquieren del entorno.

Para poder configurar los esclavos de una manera más fácil sin tener que usar el programa XCTU y no tener que introducir los datos de configuración uno a uno se lleva a cabo el

desarrollo de una aplicación de Windows cuyo contenido sea una interfaz que unifique todos los pasos de configuración de los esclavos en sí misma.

Para ello, se escoge el entorno de desarrollo integrado Microsoft Visual Studio, el cual facilita la tarea de la creación de aplicaciones para Windows gracias a su funcionalidad Windows Forms.

En Windows Forms se hace uso de las librerías .NET Framework. Además, se utilizan ficheros de extensión .cs y se programa en un lenguaje que nos será de gran uso y que suele estar orientado a la programación de objetos: el C#.

Windows Forms se basa principalmente en la creación de formularios, que son superficies visuales en las que el cliente puede observar una determinada información. Las aplicaciones de Windows Forms se implementan añadiendo controles a los formularios y programando cómo responde la aplicación a cada una de las acciones de usuario (eventos), tales como hacer click en un botón o pulsar una determinada tecla.

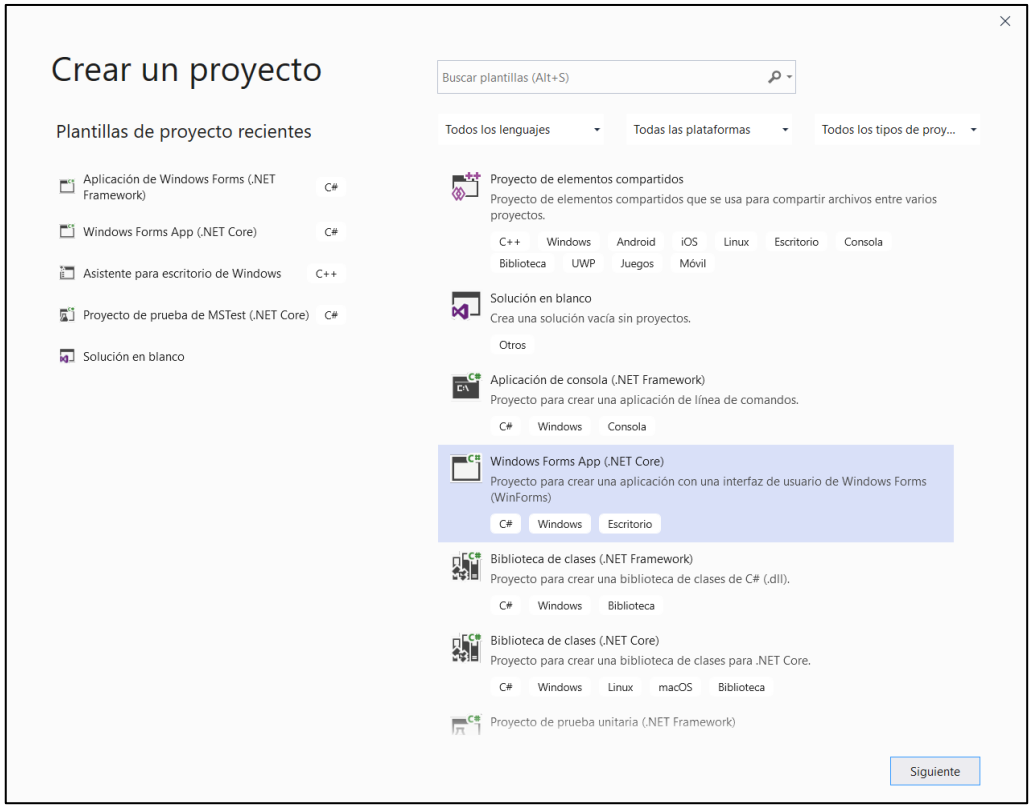

Figura 4-10: Creación del proyecto Windows Forms en Visual Studio.

La interfaz está formada por ocho formularios de Windows:

- Un menú principal: es la primera pantalla que se abre cuando se ejecuta el programa.
- Seis formularios distintos a los que se accede desde el menú principal, uno para cada posible número de esclavos a configurar (3 filas, 4 filas, 5 filas, 6 filas, 7 filas y 8 filas.)
- Una pantalla de “ayuda” y otra de “acerca de” accesibles desde el menú principal.

En Microsoft Visual Studio, se permite configurar cada formulario de manera visual (añadiendo elementos como botones, fondos, trackbars, etc), y también introducirnos en el código que generan los eventos de dichos elementos, como se puede ver en las siguientes figuras:

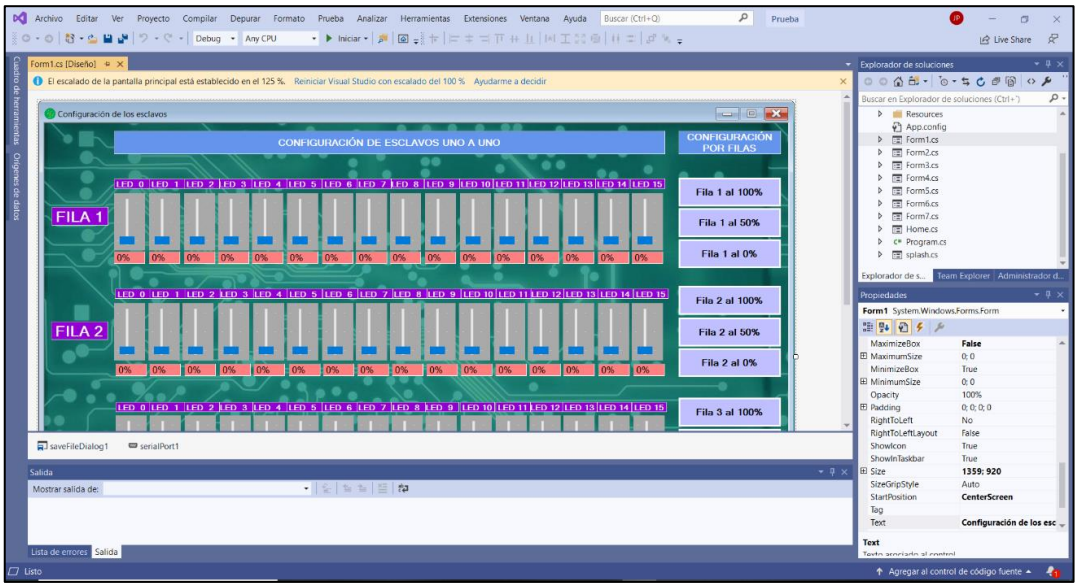

Figura 4-11: Diseño del primer formulario

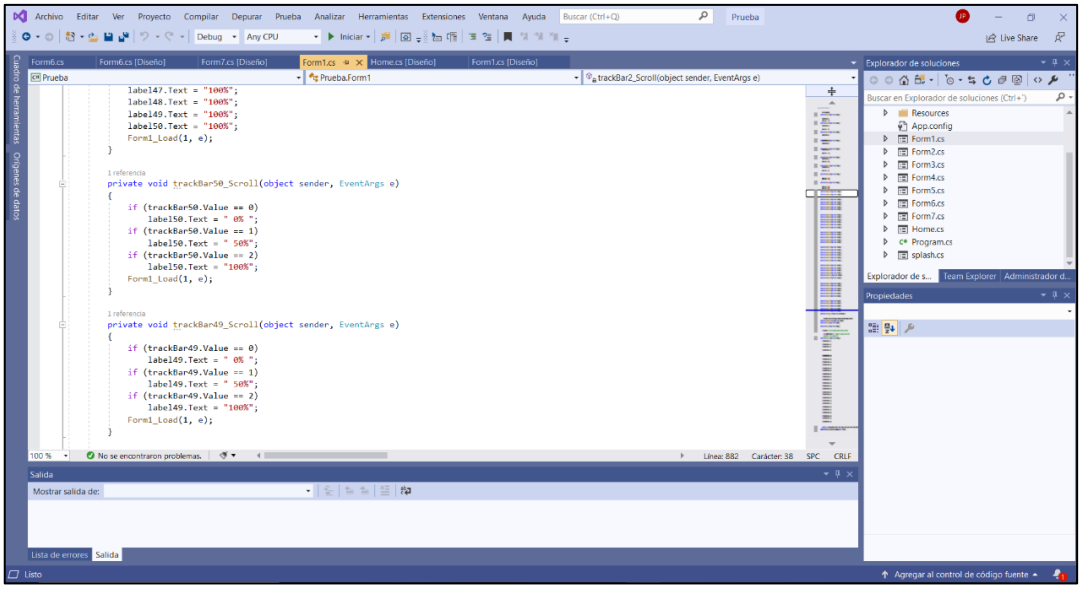

Figura 4-12: Código del primer formulario.

Dado que el número de líneas de código de cada formulario supera las 1400, se van a comentar a continuación las funciones o elementos más importantes que se han usado. Como muestra, se puede consultar el código del formulario de tres filas en el apartado B del Anexo. El código de las demás disposiciones es igual, pero añadiendo los elementos nuevos con respecto al diseño de tres filas y teniéndolos en cuenta en todas las funciones, lo cual agrega bastantes líneas de código en cada nueva fila que se incorpora.

Funciones más importantes:

- private void button_Click {}:
  Estas son las funciones de botón. Cada botón está asociado a una distinta, y dentro de ellas se establece lo que sucede cuando ocurre un evento de botón, (es decir, cuando se hace click en el botón.)
  Se han utilizado botones para numerosos usos, por ejemplo, los botones cuya función es fijar que todos los trackbars de una fila tomen un mismo valor de pwm al ser pulsados,

  En cada uno de los seis formularios de configuración, existen dos botones que adquieren una gran importancia (los que reemplazan el uso de XCTU), y que a continuación se comentará cómo se han creado:

  - Botón que establece la conexión serie: Introducimos en él el código que efectúa la apertura de la comunicación serie entre nuestro ordenador y el Xbee. Para ello, se definen las condiciones de esa comunicación por medio de los parámetros que aparecen en la siguiente figura , los cuales se consultan en XCTU.

```
//Definir los parámetros de la comunicación serial
serialPort1.PortName = comboBox1.Text;
serialPort1.BaudRate = 9600;
serialPort1.Parity = Parity.None;
serialPort1.DataBits = 8;
serialPort1.StopBits = StopBits.One;
serialPort1.Open();  //Abrir la conexión
```

Figura 4-13: Parámetros de la conexión serie.

  - Botón que envía los datos a los esclavos: Dentro de él, se declaran tantos strings como leds haya que configurar, (es decir, tantos como el número de trackbars existente), y se configura con bucles if para que cada string de un bit almacene el estado actual de cada trackbar (que esté fijado que el led luzca un 0%, un 50% o un 100%; aunque en este caso la configuración se debe hacer con caracteres ASCII). Una vez esos strings se rellenan correctamente, se utiliza la función string.concat para concatenarlos todos en orden en un único string el cual se envía al maestro para que maneje los esclavos mediante la función serialPort1.Write();

Decimal	Hexadecimal	Símbolo ASCII
0	00	\0
50	32	2
100	64	d

Tabla 4-2: Tabla de valores usados en ASCII.

- private void trackBar_Scroll {}:

  Estas son las funciones de trackbar o barra de desplazamiento. Cada trackbar está asociado a una de estas funciones, y a su vez en este caso, controla el nivel de pwm que se le fija a cada led por separado. Dentro de cada función trackBar_Scroll, se define qué ocurre cuando acontece un evento de desplazamiento de la barra mediante

el uso de bucles if{}. En el interior de esos bucles if, se escriben en etiquetas (labels) los valores del trackbar que se están fijados en cada momento.
Estas etiquetas también tienen sus propias funciones de eventos, pero en este caso se quedarán vacías ya que lo único que nos interesa hacer con ellas es que cambien la información que muestran cuando se desplacen los trackbars, pero no queremos que ocurra nada cuando se haga click sobre ellas.

- ComboBox:
  Este elemento también tiene una función asociada a él, pero en este caso la parte de código se deja vacía. En cuanto a la parte de diseño, se implementa para poder seleccionar el puerto al que se ha conectado el XBee y por lo tanto, con el que se quiere establecer la conexión serie.

- SerialPort:
  Este elemento no tiene ninguna función asociada a él, simplemente se añade a la pestaña de diseño del formulario y se configuran sus parámetros en el display de la siguiente forma:

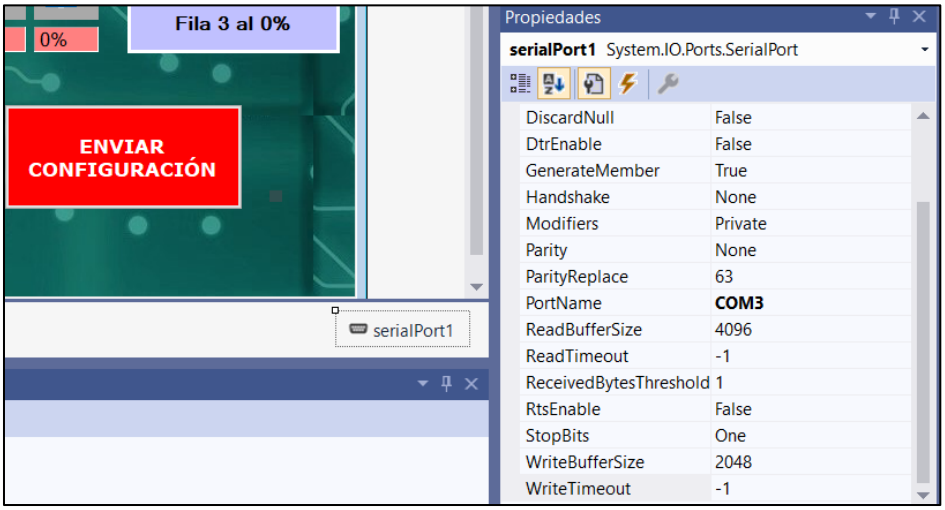

Figura 4-14: Parámetros del elemento SerialPort.

5 Integración, pruebas y resultados

Dando ya por finalizadas las fases de diseño y desarrollo del TFG, da comienzo la fase de integración, pruebas y resultados; en la que se comprobará que todo lo desarrollado funcione correctamente.

5.1 Prueba de la comunicación inalámbrica.

La primera prueba que se va a realizar es comprobar si el dispositivo funciona de forma correcta, todavía sin introducir la aplicación de Windows, es decir, configurando los esclavos a mano.

Para ello, se conecta por el puerto USB al ordenador el dispositivo XBee coordinador y se abre el programa XCTU. Como ya se han configurado los parámetros de la comunicación inalámbrica en el apartado 4.1, después de buscar y añadir el dispositivo XBee, se va directamente al modo Consola.

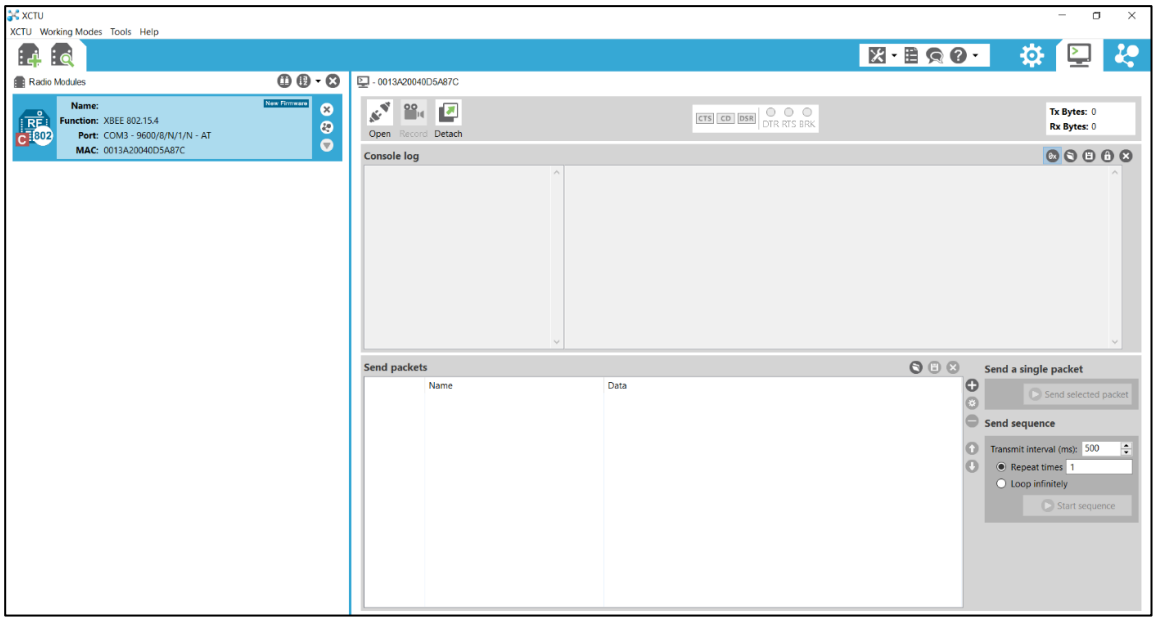

Figura 5-1: Modo Consola en XCTU.

Se selecciona la opción de abrir comunicación inalámbrica (Open) y se conecta por otro puerto USB el PCB que contiene al dispositivo XBee receptor y la plataforma Teensy.

A su vez, este PCB se conecta por medio de su conector I2C a los PCBs de material flexible, es decir, a los esclavos de la comunicación I2C.

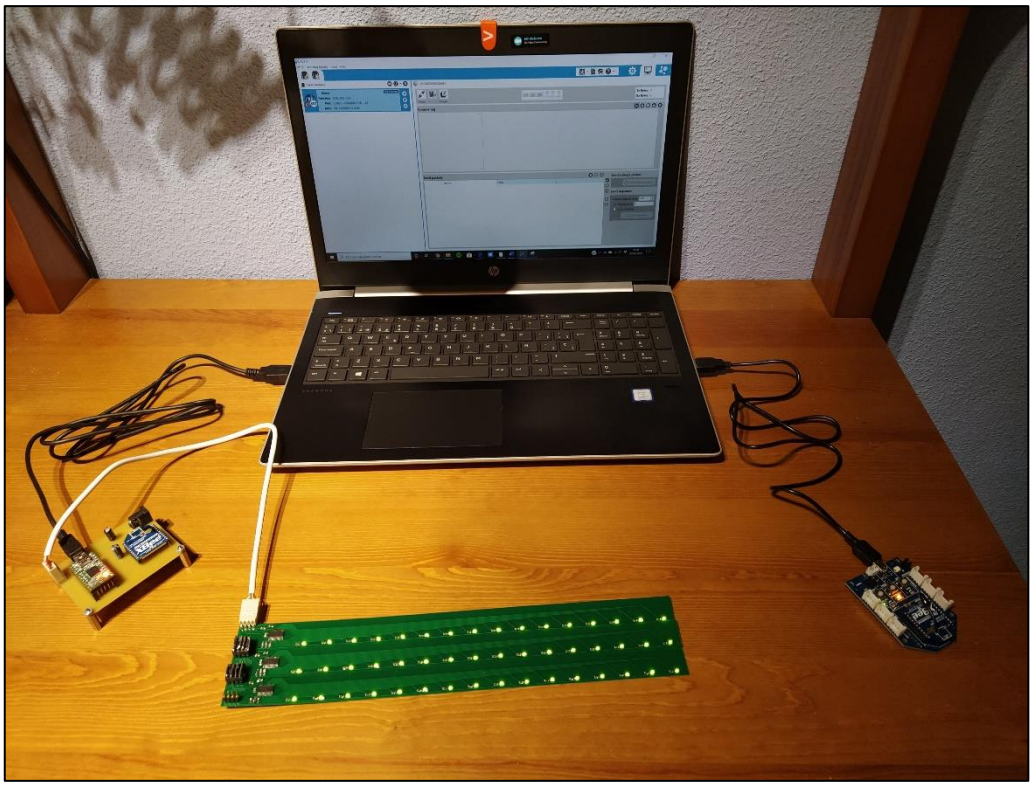

Figura 5-2: Montaje de todos los dispositivos.

A continuación, se abre la aplicación teensy.exe, desde la cual se baja a placa el fichero .hex que se ha generado en AtmelStudios en el apartado 4.3.

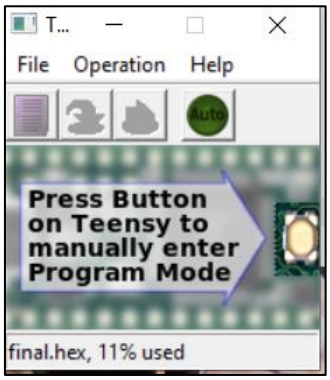

Figura 5-3: Teensy Loader.

Cuando ya se ha programado la plataforma Teensy, se vuelve a XCTU.
En XCTU aparecerán por consola los mensajes que indican que ya podemos enviar los datos de configuración, (“Listo para la comunicación”).
Para ello, se hace click en “Add new packet to the list”. Ese paquete serán los datos de configuración de los esclavos, y se pueden escribir tanto como símbolos ASCII como en sistema hexadecimal. En este caso, se opta por el sistema hexadecimal.
Debido a que contamos con tres filas de esclavos, y cada fila posee 16 diodos led a configurar, tendremos que enviar un paquete de 48 Bytes.

Por ejemplo, lo configuraremos para que la primera fila de leds luzca al 50 % (valor 32 en hexa), la segunda al 0% (00 en hexa), y la tercera fila luzca al 100% (valor 64 en Hexa.)

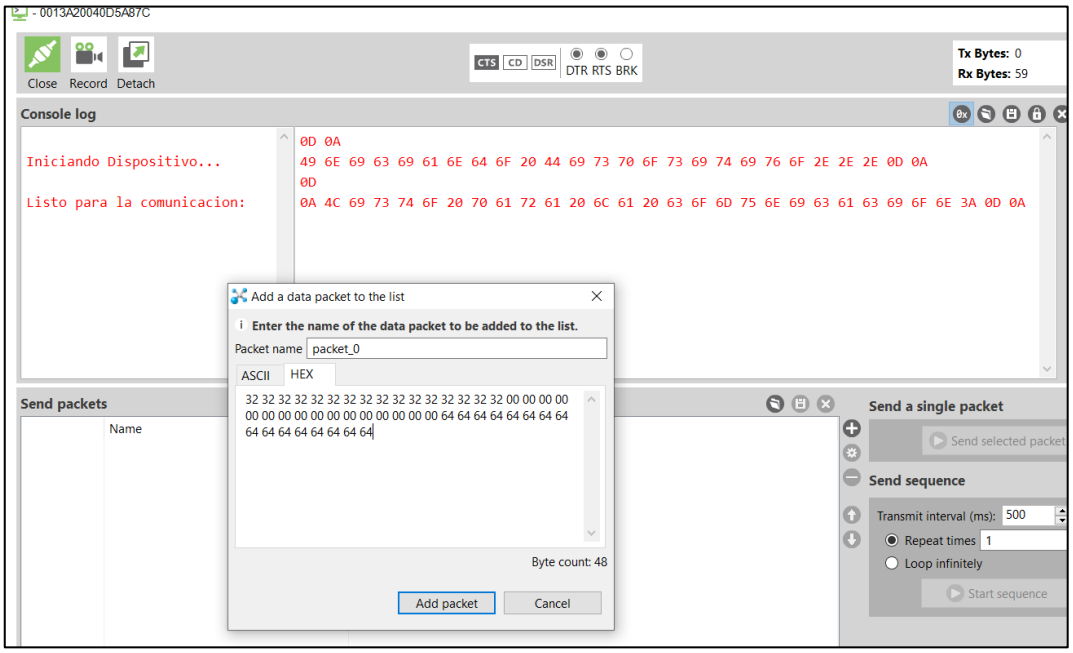

Figura 5-4: Creación de paquete a enviar en XCTU.

Definidos ya los datos a enviar, se hace click en Add Packet y seguidamente en Send selected packet.

Como se puede observar en la siguiente fotografía, la prueba se ha desarrollado correctamente, siendo el resultado el esperado.

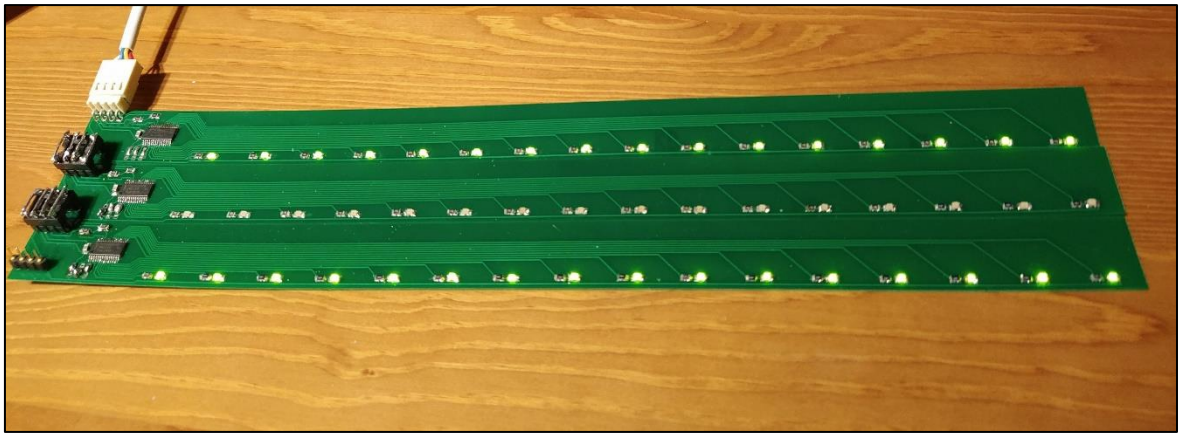

Figura 5-5: Fotografía del estado final de los esclavos.

5.2 Prueba del sistema completo.

Esta segunda y última prueba es igual que la anterior excepto en que, esta vez sí, se utilizará la aplicación Windows que se ha desarrollado con el fin de configurar los esclavos y realizar la conexión serie desde ella, evitando así el uso de XCTU.

Se colocan todos los dispositivos de la misma forma que en la prueba anterior:
Se conectan por medio de dos puertos USB distintos el dispositivo XBee coordinador y el PCB que integra el receptor de radio y la plataforma Teensy. A dicho PCB, se le conectan los PCBs de material flexible por medio de I2C. (Mismo montaje que en el anterior apartado, figura 5-2).

Se baja el fichero .hex de la misma forma que en la anterior prueba, y se abre la aplicación que se ha desarrollado.

Al abrirla, nos encontramos con un menú principal desde el que se puede acceder por medio de botones a los demás formularios.

Figura 5-6: Menú principal de la aplicación.

Este menú tiene la posibilidad de abrir un formulario en el que se explica cómo utilizar la aplicación (AYUDA) y otro que describe su perspectiva o contexto general (ACERCA DE.)

Debajo del mensaje de bienvenida, se nos permite elegir qué disposición de filas se va a utilizar. En el caso de este TFG, se utilizan tres filas de leds, pero la aplicación está preparada para ser perfectamente funcional con hasta ocho filas (128 motores o leds) a configurar.

Cuando se hace click en el botón de tres filas, aparece la siguiente interfaz:

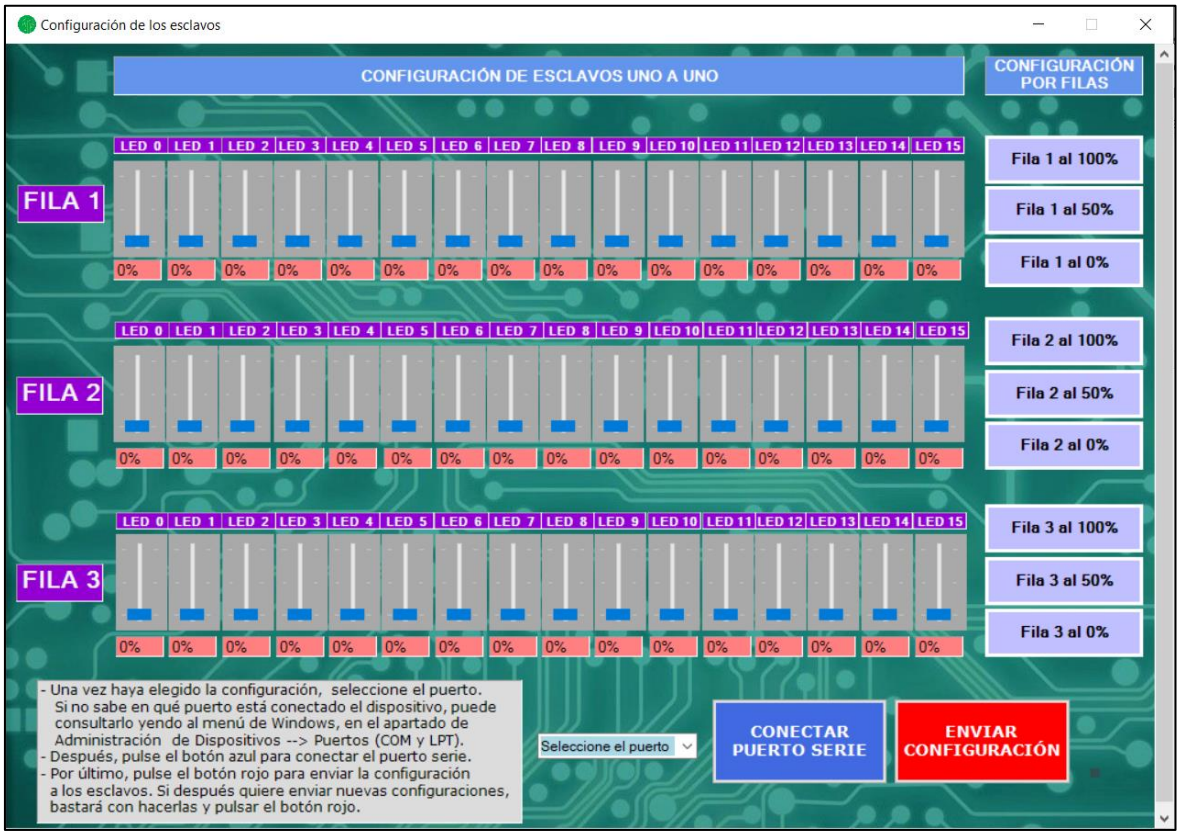

Figura 5-7: Interfaz de configuración para tres filas.

En ella, nos encontramos con botones que permiten configurar cada fila de forma completa, y también unas barras de desplazamiento que permiten configurar cada led de cada fila.

Una vez se ha elegido la configuración deseada, se selecciona el puerto al que está conectado el dispositivo coordinador XBee y se pulsa el botón azul, el cual establece la conexión con el puerto serie.

Por último, se hace click en el botón rojo para enviar los datos de la configuración elegida (un array con tantos símbolos ASCII como número de leds hay) al receptor de radio, el cual a su vez se comunica con el mastreo del bus I2C, que se encarga de controlar a los esclavos (los PCBs que conforman el peto.)

Para la prueba, se configurarán por ejemplo las filas de la siguiente forma, y así comprobar su correcto funcionamiento:

- Fila 1: Los leds con número par (empezando por el 0), se configurarán con un brillo al 100%, mientras que los de número impar, se configurarán al 0%.

- Fila 2: Los diodos led con número impar se configurarán con un brillo al 50% y los leds con número par, al 0%.

- Fila 3: Los seis primeros leds se configurarán con un brillo al 50%, los cinco siguientes al 0% y los cinco últimos al 100%.

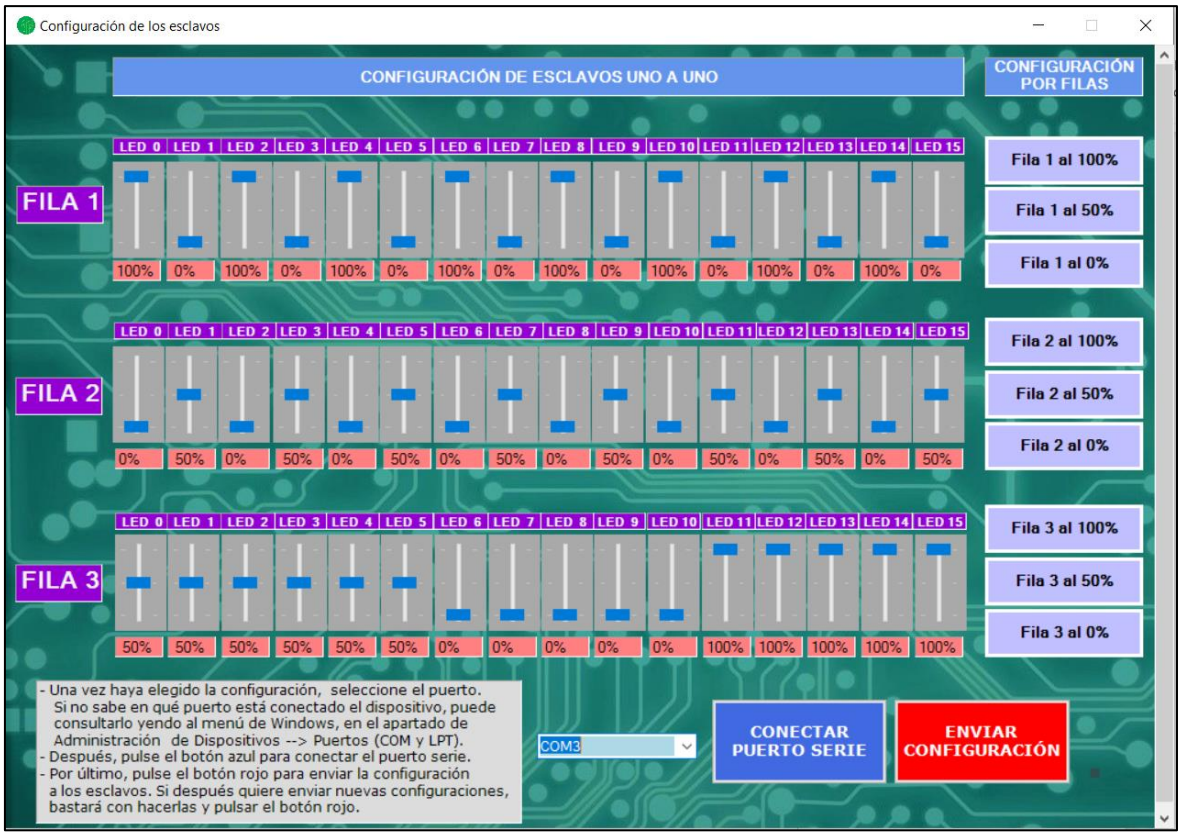

Figura 5-8: Configuración de la prueba y puertos ya seleccionados.

Como se puede observar en la siguiente figura, el resultado final es el correcto, luego podemos afirmar que esta aplicación sustiuye perfectamente la operativa que antes se realizaba con en el programa XCTU, haciendo más fáciles y rápidas las pruebas que se quieran realizar con este prototipo.

Para el dispositivo final, recordemos que se sustituirán los diodos led por motores vibratorios, que serán los que verdaderamente dotarán al chaleco de la usabilidad deseada para personas con discapacidad visual.

Figura 5-9: Fotografía del estado final de los leds para la prueba.

6 Conclusiones y trabajo futuro

6.1 Conclusiones

Al finalizar este TFG, se han conseguido sus principales objetivos, los cuales eran la construcción del prototipo diseñado en [16] con PCBs de material flexible (atribuyendo a cada uno de ellos una dirección y soldando todos sus componentes), la adaptación del código en C para que se pudiesen controlar esos PCBs (ya que el código anterior solo estaba preparado para controlar un único PCB), y el desarrollo de una aplicación de Windows que facilitase la tarea de configurar todo el proceso de comunicación por radio y la configuración de los esclavos para la posterior comunicación I2C maestro-esclavos. En el caso de la aplicación, hay que recordar que su desarrollo únicamente tiene sentido para hacer pruebas con el sistema, pero en realidad en el diseño final no se utilizará, puesto que el valor de vibración se enviará a cada motor mediante la interfaz encargada del tratamiento de los datos que se adquieran del entorno.

En resumen, se ha conseguido construir la última parte del prototipo desarrollado por los investigadores del Laboratorio de Visión y Percepción de la Facultad de Psicología de la UAM, es decir, la que abarcaría desde el peto hasta el procesamiento de la información en el ordenador. Con lo cual, para conseguir el prototipo final y poder empezar a investigar con él, únicamente sería necesario sustituir los diodos led por motores, y una vez realizado ese cambio, todo el hardware y el software estarían listos para darles el uso para el que fueron desarrollados: ser un dispositivo de sustitución sensorial orientado a personas con discapacidad visual.

6.2 Trabajo futuro

Como trabajo de futuro, ya se podrían realizar pruebas contando con todo el prototipo entero (el sistema de adquisición y tratamiento de imágenes), y construir los cinco PCBs de material flexible restantes para contar con el peto completo y comenzar a hacer pruebas con él. Para ello, simplemente habría que sustituir los diodos led por los motores.

Además, la aplicación cuenta con la posibilidad de configurar los leds a tres intensidades distintas (0%, 50% y 100%), lo cual parece suficiente para realizar pruebas con este prototipo, pero si en algún momento se necesitase poder configurar cualquier valor comprendido entre 0 y 100, podría hacerse al tratarse de un PWM. Lo único que se tendría que hacer es sustituir los trackbars de cada led por cuadros de texto en los que se escribiría el valor de la intensidad a la que se querría hacer brillar cada led, e introducir la configuración de ese evento en el código del botón encargado de enviar la configuración. Es importante recordar que los datos se envían en ASCII.

También, a partir del diseño de los PCBs de material flexible actual en Altium en el que los pines A5, A4 y A3 del chip PCA9685 están prefijados a 0 (GND), se podrían diseñar para que puedan ser configurables también como en el caso de los pines A2, A1 y A0. Esto permitiría poner hasta 64 filas de 16 motores cada fila, es decir, 1024 motores. Esto último sería realizable debido a que cada fila ocupa una única dirección de esclavo, y el I2C permite hasta 128 direcciones, (de 0 a 127).

Referencias

[1] Bourne RRA, Flaxman SR, Braithwaite T, Cicinelli MV, Das A, Jonas JB, et al.; Vision Loss Expert Group. Magnitude, temporal trends, and projections of the global prevalence of blindness and distance and near vision impairment: a systematic review and meta-analysis. Lancet Glob Health. 2017 Sep;5(9):e888–97.

[2] Fricke, TR, Tahhan N, Resnikoff S, Papas E, Burnett A, Suit MH, Naduvilath T, Naidoo K, Global Prevalence of Presbyopia and Vision Impairment from Uncorrected Presbyopia: Systematic Review, Meta-analysis, and Modelling, Ophthalmology. 2018 May 9

[3] P. Bach-y-Rita (1969), "Tactile sensory substitution studies", Annals of the New York Academy of Sciences, 1013: 83-91.

[4] Charles Lenay, Olivier Gapenne, Sylvain Hanneton, Catherine Marque, Christelle Genouëlle. SENSORY SUBSTITUTION: LIMITS AND PERSPECTIVES. Yvette Hatwell; Arlette Streri; Edouard Gentaz. Touching for Knowing, 53, John Benjamins Publishers, pp.275-292, 2004, Advances in Consciousness Research, 9789027251855. ff10.1075/aicr.53.22lenff. ffhal-02434266fl

[5] Schmidmaier, Matthias. (2020). Sensory Substitution Systems.

[6] Brill, J. & Lawson, Ben & Rupert, Angus. (2014). Tactile Situation Awareness System (TSAS) as a Compensatory Aid for Sensory Loss. Proceedings of the Human Factors and Ergonomics Society Annual Meeting. 58. 1028-1032. 10.1177/1541931214581215.

[7] Alvaro Cassinelli, Carson Reynolds & Masatoshi Ishikawa; The Haptic Radar/ Extended Skin Project [Wearable laser-based whiskers for augmented spatial awareness], Ishikawa-Namiki-Komuro Laboratory, University of Tokyo ,2006.

[8] S. Shoval, J. Borenstein, and Y. Koren, "Mobile robot obstacle avoidance in a computerized travel aid for the blind," in Proc. 1994 IEEE Robot. Autom. Conf., San Diego, CA, May 8–13, pp. 2023–2029.

[9] L. A. Jones, N. B. Sarter, "Tactile Displays: Guidance for Their Design and Application", HUMAN FACTORS, Vol. 50, No. 1, February 2008, pp. 90–111.

[10] R. Audette, J. Balthazaar, C. Dunk, J. Zelek, "A stereo-vision system for the visually impaired," Sch. Eng., Univ. Guelph, Guelph, ON, Canada, Tech. Rep. 2000-41x-1, 2000.

[11] S. Meers, K. Ward, "A substitute vision system for providing 3D perception and GPS navigation via electro-tactile stimulation," presented at the 1st Int. Conf. Sens. Technol., Palmerston North, New Zealand, Nov. 21–23, 2005.

[12] L. Cancar, A. Díaz, A. Barrientos, D. Travieso y D. M. Jacobs, "Tactile-Sight: A Sensory Substitution Device Base on Distance-Related Vibrotactile Flow", International Journal of Advanced Robotic Systems, Regular Paper, Madrid, Febrero de 2013.

[13] Digi International, XBee/XBee-PRO S1 802.15.4 (Legacy).
[14] Atmel, ATmega32U4: 8-bit AVR Microcontroller with 32K Bytes of ISP Flash and USB Controller.
[15] NXP, PCA9685 16-channel, 12-bit PWM Fm+ I2C-bus LED controller.
[16] José Antonio Torre Albarsanz, Trabajo de Fin de Máster, “Sustitución sensorial con motores de vibración", septiembre de 2018, UAM.

Glosario

SSD	Sensory Substitution Device
PCB	Printed Circuit Board
ASCII	American Standard Code for Information Interchange
LED	Light Emitting Diode
I2C	Inter-Integrated Circuit
PWM	Pulse Width Modulation

Anexos

A Código desarrollado para control de matriz de motores.

```
/*****************************************************************
** ARCHIVO: main.c                                              **
**                                                              **
**                                                              **
** Archivo que contiene el codigo principal del software.       **
**                                                              **
**                                                              **
** AUTOR: Javier de Pedro Pascual (modificando código original de
José Antonio Torre)                                            **
**                                                              **
** FECHA: 04/02/2020                                            **
**                                                              **
**                                                              **
*****************************************************************/

#include <avr/io.h>
#include "i2c.h"
#include "Xbee.h"
#include <util/delay.h>
#include "PCA9685.h"
#include <stdio.h>
#include <string.h>
#include <stdlib.h>


//Función de encendido de los esclavos

void encendido(int dimension, uint8_t addrMaster1){
      for(int j = 0; j < dimension; j++){
            setPWM(addrMaster1, j, 4096, 0);
      }

      for(int times = 0; times < 1; times++){
            for (int j = 0; j < dimension; j++){
                  if (j==0){
                        setPWM(addrMaster1, j, 0, 0);
                        setPWM(addrMaster1, j+1, 4096, 0);
                  }
                  else{
                        setPWM(addrMaster1, j-1, 4096, 0);
                        setPWM(addrMaster1, j, 0, 0);
                        setPWM(addrMaster1, j+1, 4096, 0);
                  }
                  _delay_ms(50);
```

```
for (int i = 0; i < dimension; i++){
                    if (i==15){
                          setPWM(addrMaster1, 15-i, 0, 0);
                          setPWM(addrMaster1, 15-i+1, 4096, 0);
                    }
                    else{
                          setPWM(addrMaster1, 15-i-1, 4096, 0);
                          setPWM(addrMaster1, 15-i, 0, 0);
                          setPWM(addrMaster1, 15-i+1, 4096, 0);
                    }
                    _delay_ms(50);
              }
       }

       for(int j = 0; j < dimension; j++){
             setPWM(addrMaster1, j, 4096, 0);
       }
}

int main(void)
{
       //ENCENDIDO DEL LED DEL MICRO.
       LED_CONFIG;
       LED_ON;

       //INICIALIZACIÒN DE DIRECCIONES DE LOS ESCLAVOS.
       uint8_t addrMaster1 = 0x40;   //PRIMER ESCLAVO

       uint8_t addrMaster2 = 0x41;   //SEGUNDO ESCLAVO

       uint8_t addrMaster3 = 0x42;       //TERCER ESCLAVO

     // (DESCOMENTAR CUANDO SE AÑADAN MÀS ESCLAVOS)
       //uint8_t addrMaster4 = 0x43;       //CUARTO ESCLAVO
       //uint8_t addrMaster5 = 0x44;       //QUINTO ESCLAVO
       //uint8_t addrMaster6 = 0x45;       //SEXTO ESCLAVO
       //uint8_t addrMaster7 = 0x46;       //SÈPTIMO ESCLAVO
       //uint8_t addrMaster8 = 0x47;       //OCTAVO ESCLAVO

       //INICIALIZACIÒN DEL DRIVER I2C:

       //PRIMER ESCLAVO
       begin(addrMaster1);
       setPWMFreq(addrMaster1, 1600);

       //SEGUNDO ESCLAVO
       begin(addrMaster2);
       setPWMFreq(addrMaster2, 1600);

       //TERCER ESCLAVO
       begin(addrMaster3);
       setPWMFreq(addrMaster3, 1600);


       //(DESCOMENTAR CUANDO SE AÑADAN MÀS ESCLAVOS)

       //CUARTO ESCLAVO
       //begin(addrMaster4);
       //setPWMFreq(addrMaster4, 1600);
```

```
      //QUINTO ESCLAVO
      //begin(addrMaster5);
      //setPWMFreq(addrMaster5, 1600);

      //SEXTO ESCLAVO
      //begin(addrMaster6);
      //setPWMFreq(addrMaster6, 1600);

      //SÈPTIMO ESCLAVO
      //begin(addrMaster7);
      //setPWMFreq(addrMaster7, 1600);

      //OCTAVO ESCLAVO
      //begin(addrMaster8);
      //setPWMFreq(addrMaster8, 1600);

      //INICIALIZACION VARIABLES.
      int dimension = 48; //Sumarle 16 con cada fila de más que
se añada.
      int dimension_1 = 16;

      uint8_t array[dimension];
      //Se declaran arrays para manejo de las tres filas ////
      uint8_t array1[dimension_1];
      uint8_t array2[dimension_1];
      uint8_t array3[dimension_1];

      //(DESCOMENTAR CUANDO SE AÑADAN MÀS ESCLAVOS)

      //uint8_t array4[dimension_1];
      //uint8_t array5[dimension_1];
      //uint8_t array6[dimension_1];
      //uint8_t array7[dimension_1];
      //uint8_t array8[dimension_1];

      int valor_a = 0;
      int valor_b = 0;
      int valor_c = 0;

      //(DESCOMENTAR CUANDO SE AÑADAN MÀS ESCLAVOS)

      //int valor_d = 0;
      //int valor_e = 0;
      //int valor_f = 0;
      //int valor_g = 0;
      //int valor_h = 0;


      int pwm_a, pwm_b, pwm_c;

      //(DESCOMENTAR CUANDO SE AÑADAN MÀS ESCLAVOS)
      //int pwm_d, pwm_e, pwm_f, pwm_g, pwm_h;

      ////////////////////////////////////////////////

      //INICIALIZACIÓN DEL XBEE:
      CPU_PRESCALE(0);  // run at 16 MHz
      uart_init(BAUD_RATE);
```

```
//SECUENCIA DE ENCENDIDO.
      uart_print("\r\nIniciando Dispositivo...\r\n");
      encendido(dimension_1, addrMaster1);
      encendido(dimension_1, addrMaster2);
      encendido(dimension_1, addrMaster3);

      //(DESCOMENTAR CUANDO SE AÑADAN MÀS ESCLAVOS)
      //encendido(dimension_1, addrMaster4);
      //encendido(dimension_1, addrMaster5);
      //encendido(dimension_1, addrMaster6);
      //encendido(dimension_1, addrMaster7);
      //encendido(dimension_1, addrMaster8);
      uart_print("\r\nListo para la comunicacion:\r\n");

      while(1){

            if (uart_available() >= dimension) {
                  // ALmacenamos la totalidad de los datos
                  for(int i = 0; i < dimension; i++){
                        array[i] = uart_getchar();
                  }
                  //Se asocia un array a cada fila de leds //
                  for(int i = 0; i < dimension_1; i++){
                        array1[i] = array[i];
                  }
                  for(int i = 0; i < dimension_1; i++){
                        array2[i] = array[i+dimension_1];
                  }
                  for(int i = 0; i < dimension_1; i++){
                        array3[i] = array[i+dimension_1*2];
                  }

                  //(DESCOMENTAR CUANDO SE AÑADAN MÀS ESCLAVOS)
                  //for(int i = 0; i < dimension_1; i++){
                        //array4[i] = array[i+dimension_1*3];
                  //}
                  //for(int i = 0; i < dimension_1; i++){
                        //array5[i] = array[i+dimension_1*4];
                  //}
                  //for(int i = 0; i < dimension_1; i++){
                        //array6[i] = array[i+dimension_1*5];
                  //}
                  //for(int i = 0; i < dimension_1; i++){
                        //array7[i] = array[i+dimension_1*6];
                  //}
                  //for(int i = 0; i < dimension_1; i++){
                        //array8[i] = array[i+dimension_1*7];
                  //}

      ////////////////////////////////////////////////

                  int pin = 0;
                  for(int i=0; i < dimension_1; i++){
                        int a, b, c;
      //cálculo de valor pwm para cada fila por separado////
                        a = array1[i];
                        valor_a = 100 - a;
                        pwm_a = floor(valor_a*40.95);

                        b = array2[i];
```

```
                        valor_b = 100 - b;
                        pwm_b = floor(valor_b*40.95);

                        c = array3[i];
                        valor_c = 100 - c;
                        pwm_c = floor(valor_c*40.95);

      ////////////////////////////////////////////////////////

                  //Si la variable pwm es 0, se apaga el led////

                        if(pwm_a == 0){
                              setPWM(addrMaster1, pin, 4095, 0);
                        }

                        if(pwm_b == 0){
                              setPWM(addrMaster2, pin, 4095, 0);
                        }

                        if(pwm_c == 0){
                              setPWM(addrMaster3, pin, 4095, 0);
                        }



                  //(DESCOMENTAR CUANDO SE AÑADAN MÁS ESCLAVOS)

                        //if(pwm_d == 0){
                              //setPWM(addrMaster4, pin, 4095,0);
                        //}
                        //if(pwm_e == 0){
                              //setPWM(addrMaster5, pin, 4095,0);
                        //}
                        //if(pwm_f == 0){
                              //setPWM(addrMaster6, pin, 4095,0);
                        //}
                        //if(pwm_g == 0){
                              //setPWM(addrMaster7, pin, 4095,0);
                        //}
                        //if(pwm_h == 0){
                              //setPWM(addrMaster8, pin, 4095,0);
                        //}



                        if (pwm_a != 0){
                              setPWM(addrMaster1, pin, 0, pwm_a);

                        }

                        if (pwm_b != 0){
                              setPWM(addrMaster2, pin, 0, pwm_b);
                        }

                        if (pwm_c != 0){
                              setPWM(addrMaster3, pin, 0, pwm_c);
                        }
```

```
                  //(DESCOMENTAR CUANDO SE AÑADAN MÁS ESCLAVOS)
                        //if (pwm_d != 0){
                              //setPWM(addrMaster4, pin, 0,
pwm_d);
                        //}
                        //if (pwm_e != 0){
                              //setPWM(addrMaster5, pin, 0,
pwm_e);
                        //}
                        //if (pwm_f != 0){
                              //setPWM(addrMaster6, pin, 0,
pwm_f);
                        //}
                        //if (pwm_g != 0){
                              //setPWM(addrMaster7, pin, 0,
pwm_g);
                        //}
                        //if (pwm_h != 0){
                              //setPWM(addrMaster8, pin, 0,
pwm_h);
                        //}


      ////////////////////////////////////////////////////7/////
                        pin++;
                  }
                  uart_print(" Fin\r\n");
            }
      }
}
```

B Código de la interfaz para 3 filas de motores en Visual Studio.

```
/*****************************************************************************************************
** ARCHIVO: Form1.cs                                                                                **
**                                                                                                  **
** Archivo que contiene el codigo del primer formulario.                                            **
**                                                                                                  **
** AUTOR: Javier de Pedro Pascual                                                                   **
** FECHA: 04/02/2020                                                                                **
**                                                                                                  **
*****************************************************************************************************/
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Threading;
using System.IO.Ports;


namespace Prueba
{
    public partial class Form1 : Form
    {

        public Form1()
        {
            InitializeComponent();
        }

        bool conectado = false;

        private void Form1_Load(object sender, EventArgs e)
        {

        }

        private void button3_Click(object sender, EventArgs e)
        {
            trackBar1.Value = 1;
            trackBar2.Value = 1;
            trackBar3.Value = 1;
            trackBar4.Value = 1;
            trackBar5.Value = 1;
            trackBar6.Value = 1;
            trackBar7.Value = 1;
            trackBar8.Value = 1;
            trackBar9.Value = 1;
            trackBar10.Value = 1;
            trackBar11.Value = 1;
```

```
        trackBar12.Value = 1;
        trackBar13.Value = 1;
        trackBar14.Value = 1;
        trackBar15.Value = 1;
        trackBar16.Value = 1;
        label1.Text = " 50%";
        label2.Text = " 50%";
        label3.Text = " 50%";
        label4.Text = " 50%";
        label5.Text = " 50%";
        label6.Text = " 50%";
        label7.Text = " 50%";
        label8.Text = " 50%";
        label9.Text = " 50%";
        label10.Text = " 50%";
        label11.Text = " 50%";
        label12.Text = " 50%";
        label13.Text = " 50%";
        label14.Text = " 50%";
        label15.Text = " 50%";
        label16.Text = " 50%";
        Form1_Load(1, e);
    }

    private void button4_Click(object sender, EventArgs e)
    {
        trackBar1.Value = 0;
        trackBar2.Value = 0;
        trackBar3.Value = 0;
        trackBar4.Value = 0;
        trackBar5.Value = 0;
        trackBar6.Value = 0;
        trackBar7.Value = 0;
        trackBar8.Value = 0;
        trackBar9.Value = 0;
        trackBar10.Value = 0;
        trackBar11.Value = 0;
        trackBar12.Value = 0;
        trackBar13.Value = 0;
        trackBar14.Value = 0;
        trackBar15.Value = 0;
        trackBar16.Value = 0;
        label1.Text = " 0%";
        label2.Text = " 0%";
        label3.Text = " 0%";
        label4.Text = " 0%";
        label5.Text = " 0%";
        label6.Text = " 0%";
        label7.Text = " 0%";
        label8.Text = " 0%";
        label9.Text = " 0%";
```

```
        label10.Text = " 0%";
        label11.Text = " 0%";
        label12.Text = " 0%";
        label13.Text = " 0%";
        label14.Text = " 0%";
        label15.Text = " 0%";
        label16.Text = " 0%";
        Form1_Load(1, e);
    }

    private void button5_Click(object sender, EventArgs e)
    {
        trackBar1.Value = 2;
        trackBar2.Value = 2;
        trackBar3.Value = 2;
        trackBar4.Value = 2;
        trackBar5.Value = 2;
        trackBar6.Value = 2;
        trackBar7.Value = 2;
        trackBar8.Value = 2;
        trackBar9.Value = 2;
        trackBar10.Value = 2;
        trackBar11.Value = 2;
        trackBar12.Value = 2;
        trackBar13.Value = 2;
        trackBar14.Value = 2;
        trackBar15.Value = 2;
        trackBar16.Value = 2;
        label1.Text = "100%";
        label2.Text = "100%";
        label3.Text = "100%";
        label4.Text = "100%";
        label5.Text = "100%";
        label6.Text = "100%";
        label7.Text = "100%";
        label8.Text = "100%";
        label9.Text = "100%";
        label10.Text = "100%";
        label11.Text = "100%";
        label12.Text = "100%";
        label13.Text = "100%";
        label14.Text = "100%";
        label15.Text = "100%";
        label16.Text = "100%";
        Form1_Load(1, e);
    }
```

```
private void button6_Click(object sender, EventArgs e)
{
    trackBar18.Value = 0;
    trackBar19.Value = 0;
    trackBar20.Value = 0;
    trackBar21.Value = 0;
    trackBar22.Value = 0;
    trackBar23.Value = 0;
    trackBar24.Value = 0;
    trackBar25.Value = 0;
    trackBar26.Value = 0;
    trackBar27.Value = 0;
    trackBar28.Value = 0;
    trackBar29.Value = 0;
    trackBar30.Value = 0;
    trackBar31.Value = 0;
    trackBar32.Value = 0;
    trackBar33.Value = 0;
    label18.Text = " 0%";
    label19.Text = " 0%";
    label20.Text = " 0%";
    label21.Text = " 0%";
    label22.Text = " 0%";
    label23.Text = " 0%";
    label24.Text = " 0%";
    label25.Text = " 0%";
    label26.Text = " 0%";
    label27.Text = " 0%";
    label28.Text = " 0%";
    label29.Text = " 0%";
    label30.Text = " 0%";
    label31.Text = " 0%";
    label32.Text = " 0%";
    label33.Text = " 0%";
    Form1_Load(1, e);
}

private void button7_Click(object sender, EventArgs e)
{
    trackBar18.Value = 1;
    trackBar19.Value = 1;
    trackBar20.Value = 1;
    trackBar21.Value = 1;
    trackBar22.Value = 1;
    trackBar23.Value = 1;
    trackBar24.Value = 1;
    trackBar25.Value = 1;
    trackBar26.Value = 1;
    trackBar27.Value = 1;
    trackBar28.Value = 1;
    trackBar29.Value = 1;
    trackBar30.Value = 1;
```

```
            trackBar30.Value = 1;
            trackBar31.Value = 1;
            trackBar32.Value = 1;
            trackBar33.Value = 1;
            label18.Text = " 50%";
            label19.Text = " 50%";
            label20.Text = " 50%";
            label21.Text = " 50%";
            label22.Text = " 50%";
            label23.Text = " 50%";
            label24.Text = " 50%";
            label25.Text = " 50%";
            label26.Text = " 50%";
            label27.Text = " 50%";
            label28.Text = " 50%";
            label29.Text = " 50%";
            label30.Text = " 50%";
            label31.Text = " 50%";
            label32.Text = " 50%";
            label33.Text = " 50%";
            Form1_Load(1, e);
        }

        private void button8_Click(object sender, EventArgs e)
        {
            trackBar18.Value = 2;
            trackBar19.Value = 2;
            trackBar20.Value = 2;
            trackBar21.Value = 2;
            trackBar22.Value = 2;
            trackBar23.Value = 2;
            trackBar24.Value = 2;
            trackBar25.Value = 2;
            trackBar26.Value = 2;
            trackBar27.Value = 2;
            trackBar28.Value = 2;
            trackBar29.Value = 2;
            trackBar30.Value = 2;
            trackBar31.Value = 2;
            trackBar32.Value = 2;
            trackBar33.Value = 2;
            label18.Text = "100%";
            label19.Text = "100%";
            label20.Text = "100%";
            label21.Text = "100%";
            label22.Text = "100%";
            label23.Text = "100%";
            label24.Text = "100%";
            label25.Text = "100%";
            label26.Text = "100%";
            label27.Text = "100%";
            label28.Text = "100%";
            label29.Text = "100%";
```

```
        label30.Text = "100%";
        label31.Text = "100%";
        label32.Text = "100%";
        label33.Text = "100%";
        Form1_Load(1, e);
    }

    private void button11_Click(object sender, EventArgs e)
    {
        trackBar35.Value = 0;
        trackBar36.Value = 0;
        trackBar37.Value = 0;
        trackBar38.Value = 0;
        trackBar39.Value = 0;
        trackBar40.Value = 0;
        trackBar41.Value = 0;
        trackBar42.Value = 0;
        trackBar43.Value = 0;
        trackBar44.Value = 0;
        trackBar45.Value = 0;
        trackBar46.Value = 0;
        trackBar47.Value = 0;
        trackBar48.Value = 0;
        trackBar49.Value = 0;
        trackBar50.Value = 0;
        label35.Text = " 0%";
        label36.Text = " 0%";
        label37.Text = " 0%";
        label38.Text = " 0%";
        label39.Text = " 0%";
        label40.Text = " 0%";
        label41.Text = " 0%";
        label42.Text = " 0%";
        label43.Text = " 0%";
        label44.Text = " 0%";
        label45.Text = " 0%";
        label46.Text = " 0%";
        label47.Text = " 0%";
        label48.Text = " 0%";
        label49.Text = " 0%";
        label50.Text = " 0%";
        Form1_Load(1, e);
    }

    private void button10_Click(object sender, EventArgs e)
    {
        trackBar35.Value = 1;
        trackBar36.Value = 1;
        trackBar37.Value = 1;
        trackBar38.Value = 1;
        trackBar39.Value = 1;
```

```
        trackBar40.Value = 1;
        trackBar41.Value = 1;
        trackBar42.Value = 1;
        trackBar43.Value = 1;
        trackBar44.Value = 1;
        trackBar45.Value = 1;
        trackBar46.Value = 1;
        trackBar47.Value = 1;
        trackBar48.Value = 1;
        trackBar49.Value = 1;
        trackBar50.Value = 1;
        label35.Text = " 50%";
        label36.Text = " 50%";
        label37.Text = " 50%";
        label38.Text = " 50%";
        label39.Text = " 50%";
        label40.Text = " 50%";
        label41.Text = " 50%";
        label42.Text = " 50%";
        label43.Text = " 50%";
        label44.Text = " 50%";
        label45.Text = " 50%";
        label46.Text = " 50%";
        label47.Text = " 50%";
        label48.Text = " 50%";
        label49.Text = " 50%";
        label50.Text = " 50%";
        Form1_Load(1, e);
    }

    private void button9_Click(object sender, EventArgs e)
    {
        trackBar35.Value = 2;
        trackBar36.Value = 2;
        trackBar37.Value = 2;
        trackBar38.Value = 2;
        trackBar39.Value = 2;
        trackBar40.Value = 2;
        trackBar41.Value = 2;
        trackBar42.Value = 2;
        trackBar43.Value = 2;
        trackBar44.Value = 2;
        trackBar45.Value = 2;
        trackBar46.Value = 2;
        trackBar47.Value = 2;
        trackBar48.Value = 2;
        trackBar49.Value = 2;
        trackBar50.Value = 2;
        label35.Text = "100%";
        label36.Text = "100%";
        label36.Text = "100%";
```

```
            label37.Text = "100%";
            label38.Text = "100%";
            label39.Text = "100%";
            label40.Text = "100%";
            label41.Text = "100%";
            label42.Text = "100%";
            label43.Text = "100%";
            label44.Text = "100%";
            label45.Text = "100%";
            label46.Text = "100%";
            label47.Text = "100%";
            label48.Text = "100%";
            label49.Text = "100%";
            label50.Text = "100%";
            Form1_Load(1, e);
        }

        private void trackBar50_Scroll(object sender, EventArgs e)
        {
            if (trackBar50.Value == 0)
                label50.Text = " 0% ";
            if (trackBar50.Value == 1)
                label50.Text = " 50%";
            if (trackBar50.Value == 2)
                label50.Text = "100%";
            Form1_Load(1, e);
        }

        private void trackBar49_Scroll(object sender, EventArgs e)
        {
            if (trackBar49.Value == 0)
                label49.Text = " 0% ";
            if (trackBar49.Value == 1)
                label49.Text = " 50%";
            if (trackBar49.Value == 2)
                label49.Text = "100%";
            Form1_Load(1, e);
        }

        private void trackBar48_Scroll(object sender, EventArgs e)
        {
            if (trackBar48.Value == 0)
                label48.Text = " 0% ";
            if (trackBar48.Value == 1)
                label48.Text = " 50%";
            if (trackBar48.Value == 2)
                label48.Text = "100%";
            Form1_Load(1, e);
        }
```

```
private void trackBar47_Scroll(object sender, EventArgs e)
{
    if (trackBar47.Value == 0)
        label47.Text = " 0% ";
    if (trackBar47.Value == 1)
        label47.Text = " 50%";
    if (trackBar47.Value == 2)
        label47.Text = "100%";
    Form1_Load(1, e);
}

private void trackBar46_Scroll(object sender, EventArgs e)
{
    if (trackBar46.Value == 0)
        label46.Text = " 0% ";
    if (trackBar46.Value == 1)
        label46.Text = " 50%";
    if (trackBar46.Value == 2)
        label46.Text = "100%";
    Form1_Load(1, e);
}

private void trackBar45_Scroll(object sender, EventArgs e)
{
    if (trackBar45.Value == 0)
        label45.Text = " 0% ";
    if (trackBar45.Value == 1)
        label45.Text = " 50%";
    if (trackBar45.Value == 2)
        label45.Text = "100%";
    Form1_Load(1, e);
}

private void trackBar44_Scroll(object sender, EventArgs e)
{
    if (trackBar44.Value == 0)
        label44.Text = " 0% ";
    if (trackBar44.Value == 1)
        label44.Text = " 50%";
    if (trackBar44.Value == 2)
        label44.Text = "100%";
    Form1_Load(1, e);
}

private void trackBar43_Scroll(object sender, EventArgs e)
{
    if (trackBar43.Value == 0)
        label43.Text = " 0% ";
    if (trackBar43.Value == 1)
        label43.Text = " 50%";
    if (trackBar43.Value == 2)
        label43.Text = "100%";
    Form1_Load(1, e);
}
```

```
private void trackBar42_Scroll(object sender, EventArgs e)
{
    if (trackBar42.Value == 0)
        label42.Text = " 0% ";
    if (trackBar42.Value == 1)
        label42.Text = " 50%";
    if (trackBar42.Value == 2)
        label42.Text = "100%";
    Form1_Load(1, e);
}

private void trackBar41_Scroll(object sender, EventArgs e)
{
    if (trackBar41.Value == 0)
        label41.Text = " 0% ";
    if (trackBar41.Value == 1)
        label41.Text = " 50%";
    if (trackBar41.Value == 2)
        label41.Text = "100%";
    Form1_Load(1, e);
}

private void trackBar40_Scroll(object sender, EventArgs e)
{
    if (trackBar40.Value == 0)
        label40.Text = " 0% ";
    if (trackBar40.Value == 1)
        label40.Text = " 50%";
    if (trackBar40.Value == 2)
        label40.Text = "100%";
    Form1_Load(1, e);
}

private void trackBar39_Scroll(object sender, EventArgs e)
{
    if (trackBar39.Value == 0)
        label39.Text = " 0% ";
    if (trackBar39.Value == 1)
        label39.Text = " 50%";
    if (trackBar39.Value == 2)
        label39.Text = "100%";
    Form1_Load(1, e);
}

private void trackBar38_Scroll(object sender, EventArgs e)
{
    if (trackBar38.Value == 0)
        label38.Text = " 0% ";
    if (trackBar38.Value == 1)
        label38.Text = " 50%";
    if (trackBar38.Value == 2)
        label38.Text = "100%";
    Form1_Load(1, e);
}
```

```
private void trackBar37_Scroll(object sender, EventArgs e)
{
    if (trackBar37.Value == 0)
        label37.Text = " 0% ";
    if (trackBar37.Value == 1)
        label37.Text = " 50%";
    if (trackBar37.Value == 2)
        label37.Text = "100%";
    Forml_Load(1, e);
}

private void trackBar36_Scroll(object sender, EventArgs e)
{
    if (trackBar36.Value == 0)
        label36.Text = " 0% ";
    if (trackBar36.Value == 1)
        label36.Text = " 50%";
    if (trackBar36.Value == 2)
        label36.Text = "100%";
    Forml_Load(1, e);
}

private void trackBar35_Scroll(object sender, EventArgs e)
{
    if (trackBar35.Value == 0)
        label35.Text = " 0% ";
    if (trackBar35.Value == 1)
        label35.Text = " 50%";
    if (trackBar35.Value == 2)
        label35.Text = "100%";
    Forml_Load(1, e);
}

private void trackBar33_Scroll(object sender, EventArgs e)
{
    if (trackBar33.Value == 0)
        label33.Text = " 0% ";
    if (trackBar33.Value == 1)
        label33.Text = " 50%";
    if (trackBar33.Value == 2)
        label33.Text = "100%";
    Forml_Load(1, e);
}

private void trackBar32_Scroll(object sender, EventArgs e)
{
    if (trackBar32.Value == 0)
        label32.Text = " 0% ";
    if (trackBar32.Value == 1)
        label32.Text = " 50%";
    if (trackBar32.Value == 2)
        label32.Text = "100%";
    Forml_Load(1, e);
```

```
private void trackBar31_Scroll(object sender, EventArgs e)
{
    if (trackBar31.Value == 0)
        label31.Text = " 0% ";
    if (trackBar31.Value == 1)
        label31.Text = " 50%";
    if (trackBar31.Value == 2)
        label31.Text = "100%";
    Form1_Load(1, e);
}

private void trackBar30_Scroll(object sender, EventArgs e)
{
    if (trackBar30.Value == 0)
        label30.Text = " 0% ";
    if (trackBar30.Value == 1)
        label30.Text = " 50%";
    if (trackBar30.Value == 2)
        label30.Text = "100%";
    Form1_Load(1, e);
}

private void trackBar29_Scroll(object sender, EventArgs e)
{
    if (trackBar29.Value == 0)
        label29.Text = " 0% ";
    if (trackBar29.Value == 1)
        label29.Text = " 50%";
    if (trackBar29.Value == 2)
        label29.Text = "100%";
    Form1_Load(1, e);
}

private void trackBar28_Scroll(object sender, EventArgs e)
{
    if (trackBar28.Value == 0)
        label28.Text = " 0% ";
    if (trackBar28.Value == 1)
        label28.Text = " 50%";
    if (trackBar28.Value == 2)
        label28.Text = "100%";
    Form1_Load(1, e);
}

private void trackBar27_Scroll(object sender, EventArgs e)
{
    if (trackBar27.Value == 0)
        label27.Text = " 0% ";
    if (trackBar27.Value == 1)
        label27.Text = " 50%";
    if (trackBar27.Value == 2)
        label27.Text = "100%";
    Form1_Load(1, e);
}
```

```
private void trackBar26_Scroll(object sender, EventArgs e)
{
    if (trackBar26.Value == 0)
        label26.Text = " 0% ";
    if (trackBar26.Value == 1)
        label26.Text = " 50%";
    if (trackBar26.Value == 2)
        label26.Text = "100%";
    Form1_Load(1, e);
}

private void trackBar25_Scroll(object sender, EventArgs e)
{
    if (trackBar25.Value == 0)
        label25.Text = " 0% ";
    if (trackBar25.Value == 1)
        label25.Text = " 50%";
    if (trackBar25.Value == 2)
        label25.Text = "100%";
    Form1_Load(1, e);
}

private void trackBar24_Scroll(object sender, EventArgs e)
{
    if (trackBar24.Value == 0)
        label24.Text = " 0% ";
    if (trackBar24.Value == 1)
        label24.Text = " 50%";
    if (trackBar24.Value == 2)
        label24.Text = "100%";
    Form1_Load(1, e);
}

private void trackBar23_Scroll(object sender, EventArgs e)
{
    if (trackBar23.Value == 0)
        label23.Text = " 0% ";
    if (trackBar23.Value == 1)
        label23.Text = " 50%";
    if (trackBar23.Value == 2)
        label23.Text = "100%";
    Form1_Load(1, e);
}

private void trackBar22_Scroll(object sender, EventArgs e)
{
    if (trackBar22.Value == 0)
        label22.Text = " 0% ";
    if (trackBar22.Value == 1)
        label22.Text = " 50%";
    if (trackBar22.Value == 2)
        label22.Text = "100%";
    Form1_Load(1, e);
}
```

```
private void trackBar21_Scroll(object sender, EventArgs e)
{
    if (trackBar21.Value == 0)
        label21.Text = " 0% ";
    if (trackBar21.Value == 1)
        label21.Text = " 50%";
    if (trackBar21.Value == 2)
        label21.Text = "100%";
    Form1_Load(1, e);
}

private void trackBar20_Scroll(object sender, EventArgs e)
{
    if (trackBar20.Value == 0)
        label20.Text = " 0% ";
    if (trackBar20.Value == 1)
        label20.Text = " 50%";
    if (trackBar20.Value == 2)
        label20.Text = "100%";
    Form1_Load(1, e);
}


private void trackBar19_Scroll(object sender, EventArgs e)
{
    if (trackBar19.Value == 0)
        label19.Text = " 0% ";
    if (trackBar19.Value == 1)
        label19.Text = " 50%";
    if (trackBar19.Value == 2)
        label19.Text = "100%";
    Form1_Load(1, e);
}

private void trackBar18_Scroll(object sender, EventArgs e)
{
    if (trackBar18.Value == 0)
        label18.Text = " 0% ";
    if (trackBar18.Value == 1)
        label18.Text = " 50%";
    if (trackBar18.Value == 2)
        label18.Text = "100%";
    Form1_Load(1, e);
}
```

```
private void trackBar16_Scroll(object sender, EventArgs e)
{
    if (trackBar16.Value == 0)
        label16.Text = " 0% ";
    if (trackBar16.Value == 1)
        label16.Text = " 50%";
    if (trackBar16.Value == 2)
        label16.Text = "100%";
    Form1_Load(1, e);
}

private void trackBar15_Scroll_1(object sender, EventArgs e)
{
    if (trackBar15.Value == 0)
        label15.Text = " 0% ";
    if (trackBar15.Value == 1)
        label15.Text = " 50%";
    if (trackBar15.Value == 2)
        label15.Text = "100%";
    Form1_Load(1, e);
}


private void trackBar14_Scroll(object sender, EventArgs e)
{
    if (trackBar14.Value == 0)
        label14.Text = " 0% ";
    if (trackBar14.Value == 1)
        label14.Text = " 50%";
    if (trackBar14.Value == 2)
        label14.Text = "100%";
    Form1_Load(1, e);
}

private void trackBar13_Scroll(object sender, EventArgs e)
{
    if (trackBar13.Value == 0)
        label13.Text = " 0% ";
    if (trackBar13.Value == 1)
        label13.Text = " 50%";
    if (trackBar13.Value == 2)
        label13.Text = "100%";
    Form1_Load(1, e);
}
```

```
private void trackBar12_Scroll(object sender, EventArgs e)
{
    if (trackBar12.Value == 0)
        label12.Text = " 0% ";
    if (trackBar12.Value == 1)
        label12.Text = " 50%";
    if (trackBar12.Value == 2)
        label12.Text = "100%";
    Form1_Load(1, e);
}

private void trackBar11_Scroll(object sender, EventArgs e)
{
    if (trackBar11.Value == 0)
        label11.Text = " 0% ";
    if (trackBar11.Value == 1)
        label11.Text = " 50%";
    if (trackBar11.Value == 2)
        label11.Text = "100%";
    Form1_Load(1, e);
}

private void trackBar10_Scroll(object sender, EventArgs e)
{
    if (trackBar10.Value == 0)
        label10.Text = " 0% ";
    if (trackBar10.Value == 1)
        label10.Text = " 50%";
    if (trackBar10.Value == 2)
        label10.Text = "100%";
    Form1_Load(1, e);
}

private void trackBar9_Scroll(object sender, EventArgs e)
{
    if (trackBar9.Value == 0)
        label9.Text = " 0% ";
    if (trackBar9.Value == 1)
        label9.Text = " 50%";
    if (trackBar9.Value == 2)
        label9.Text = "100%";
    Form1_Load(1, e);
}

private void trackBar8_Scroll(object sender, EventArgs e)
{
    if (trackBar8.Value == 0)
        label8.Text = " 0% ";
    if (trackBar8.Value == 1)
        label8.Text = " 50%";
    if (trackBar8.Value == 2)
        label8.Text = "100%";
    Form1_Load(1, e);
}
```

```
private void trackBar7_Scroll(object sender, EventArgs e)
{
    if (trackBar7.Value == 0)
        label7.Text = " 0% ";
    if (trackBar7.Value == 1)
        label7.Text = " 50%";
    if (trackBar7.Value == 2)
        label7.Text = "100%";
    Form1_Load(1, e);
}

private void trackBar6_Scroll(object sender, EventArgs e)
{
    if (trackBar6.Value == 0)
        label6.Text = " 0% ";
    if (trackBar6.Value == 1)
        label6.Text = " 50%";
    if (trackBar6.Value == 2)
        label6.Text = "100%";
    Form1_Load(1, e);
}

private void trackBar5_Scroll(object sender, EventArgs e)
{
    if (trackBar5.Value == 0)
        label5.Text = " 0% ";
    if (trackBar5.Value == 1)
        label5.Text = " 50%";
    if (trackBar5.Value == 2)
        label5.Text = "100%";
    Form1_Load(1, e);
}

private void trackBar4_Scroll(object sender, EventArgs e)
{
    if (trackBar4.Value == 0)
        label4.Text = " 0% ";
    if (trackBar4.Value == 1)
        label4.Text = " 50%";
    if (trackBar4.Value == 2)
        label4.Text = "100%";
    Form1_Load(1, e);
}

private void trackBar3_Scroll(object sender, EventArgs e)
{
    if (trackBar3.Value == 0)
        label3.Text = " 0% ";
    if (trackBar3.Value == 1)
        label3.Text = " 50%";
    if (trackBar3.Value == 2)
        label3.Text = "100%";
    Form1_Load(1, e);
}
```

```
private void trackBar2_Scroll(object sender, EventArgs e)
{
    if (trackBar2.Value == 0)
        label2.Text = " 0% ";
    if (trackBar2.Value == 1)
        label2.Text = " 50%";
    if (trackBar2.Value == 2)
        label2.Text = "100%";
    Form1_Load(1, e);
}

private void trackBar1_Scroll(object sender, EventArgs e)
{
    if (trackBar1.Value == 0)
        label1.Text = " 0% ";
    if (trackBar1.Value == 1)
        label1.Text = " 50%";
    if (trackBar1.Value == 2)
        label1.Text = "100%";
    Form1_Load(1, e);
}


private void saveFileDialog1_FileOk(object sender, CancelEventArgs e)
{

}

private void label1_Click(object sender, EventArgs e)
{
}


//private void button1_Click(object sender, System.EventArgs e)
//{
//    saveFileDialog1.Filter = "xml files (*.xml)|*.xml";
//    //saveFileDialog1.Filter = "txt files (*.txt)|*.txt|All files (*.*)|*.*";
//    saveFileDialog1.FilterIndex = 2;
//    saveFileDialog1.Title = "Guardar fichero de configuración";
//    saveFileDialog1.InitialDirectory = ":";
//    saveFileDialog1.FileName = "";

//    if (saveFileDialog1.ShowDialog() != DialogResult.Cancel)
//    {

//        richTextBox2.SaveFile(saveFileDialog1.FileName, RichTextBoxStreamType.PlainText);

//    }
//}
```

```
private void richTextBox1_TextChanged(object sender, EventArgs e)
{

}

private void label52_Click(object sender, EventArgs e)
{

}

private void label54_Click(object sender, EventArgs e)
{

}

private void label55_Click(object sender, EventArgs e)
{

}

private void label10_Click(object sender, EventArgs e)
{

}

private void button2_Click(object sender, EventArgs e)
{



}

private void label56_Click(object sender, EventArgs e)
{

}

private void richTextBox2_TextChanged(object sender, EventArgs e)
{

}
```

```
private void button2_Click_1(object sender, EventArgs e)
{
    if (!conectado)  //Si no esta conectado el puerto, realizar la conexión
    {
        try
        {
            //Definir los parámetros de la comunicación serial
            serialPort1.PortName = comboBox1.Text;
            serialPort1.BaudRate = 9600;
            serialPort1.Parity = Parity.None;
            serialPort1.DataBits = 8;
            serialPort1.StopBits = StopBits.One;
            serialPort1.Open();  //Abrir la conexión
            //button1.Text = "Cerrar conexión serie";
            conectado = true;
        }
        catch (ArgumentException ex)    // Código que se ejecuta en caso de error
        {
            MessageBox.Show("Error en la conexión " + ex.Message);
            serialPort1.Close(); //Cerrar la conexión
        }
        conectado = true;
    }
    else   //Cerrar conexión.
    {
        //button1.Text = "Iniciar conexión serie";
        conectado = false;
        if (serialPort1 != null)
            serialPort1.Close();
    }


}

private void button12_Click(object sender, EventArgs e)
{
    string valor1 = "\0";
    if (trackBar1.Value == 0)
        valor1 = "\0";
    if (trackBar1.Value == 1)
        valor1 = "2";
    if (trackBar1.Value == 2)
        valor1 = "d";

    string valor2 = "\0";
    if (trackBar2.Value == 0)
        valor2 = "\0";
    if (trackBar2.Value == 1)
        valor2 = "2";
    if (trackBar2.Value == 2)
        valor2 = "d";
```

```
string valor2 = "\0";
if (trackBar2.Value == 0)
    valor2 = "\0";
if (trackBar2.Value == 1)
    valor2 = "2";
if (trackBar2.Value == 2)
    valor2 = "d";

string valor3 = "\0";
if (trackBar3.Value == 0)
    valor3 = "\0";
if (trackBar3.Value == 1)
    valor3 = "2";
if (trackBar3.Value == 2)
    valor3 = "d";

string valor4 = "\0";
if (trackBar4.Value == 0)
    valor4 = "\0";
if (trackBar4.Value == 1)
    valor4 = "2";
if (trackBar4.Value == 2)
    valor4 = "d";

string valor5 = "\0";
if (trackBar5.Value == 0)
    valor5 = "\0";
if (trackBar5.Value == 1)
    valor5 = "2";
if (trackBar5.Value == 2)
    valor5 = "d";


string valor6 = "\0";
if (trackBar6.Value == 0)
    valor6 = "\0";
if (trackBar6.Value == 1)
    valor6 = "2";
if (trackBar6.Value == 2)
    valor6 = "d";

string valor7 = "\0";
if (trackBar7.Value == 0)
    valor7 = "\0";
if (trackBar7.Value == 1)
    valor7 = "2";
if (trackBar7.Value == 2)
    valor7 = "d";
```

```
string valor8 = "\0";
if (trackBar8.Value == 0)
    valor8 = "\0";
if (trackBar8.Value == 1)
    valor8 = "2";
if (trackBar8.Value == 2)
    valor8 = "d";

string valor9 = "\0";
if (trackBar9.Value == 0)
    valor9 = "\0";
if (trackBar9.Value == 1)
    valor9 = "2";
if (trackBar9.Value == 2)
    valor9 = "d";

string valor10 = "\0";
if (trackBar10.Value == 0)
    valor10 = "\0";
if (trackBar10.Value == 1)
    valor10 = "2";
if (trackBar10.Value == 2)
    valor10 = "d";

string valor11 = "\0";
if (trackBar11.Value == 0)
    valor11 = "\0";
if (trackBar11.Value == 1)
    valor11 = "2";
if (trackBar11.Value == 2)
    valor11 = "d";

string valor12 = "\0";
if (trackBar12.Value == 0)
    valor12 = "\0";
if (trackBar12.Value == 1)
    valor12 = "2";
if (trackBar12.Value == 2)
    valor12 = "d";

string valor13 = "\0";
if (trackBar13.Value == 0)
    valor13 = "\0";
if (trackBar13.Value == 1)
    valor13 = "2";
if (trackBar13.Value == 2)
    valor13 = "d";
```

```
string valor14 = "\0";
if (trackBar14.Value == 0)
    valor14 = "\0";
if (trackBar14.Value == 1)
    valor14 = "2";
if (trackBar14.Value == 2)
    valor14 = "d";

string valor15 = "\0";
if (trackBar15.Value == 0)
    valor15 = "\0";
if (trackBar15.Value == 1)
    valor15 = "2";
if (trackBar15.Value == 2)
    valor15 = "d";

string valor16 = "\0";
if (trackBar16.Value == 0)
    valor16 = "\0";
if (trackBar16.Value == 1)
    valor16 = "2";
if (trackBar16.Value == 2)
    valor16 = "d";

string valor18 = "\0";
if (trackBar18.Value == 0)
    valor18 = "\0";
if (trackBar18.Value == 1)
    valor18 = "2";
if (trackBar18.Value == 2)
    valor18 = "d";

string valor19 = "\0";
if (trackBar19.Value == 0)
    valor19 = "\0";
if (trackBar19.Value == 1)
    valor19 = "2";
if (trackBar19.Value == 2)
    valor19 = "d";

string valor20 = "\0";
if (trackBar20.Value == 0)
    valor20 = "\0";
if (trackBar20.Value == 1)
    valor20 = "2";
if (trackBar20.Value == 2)
    valor20 = "d";
```

```
string valor21 = "\0";
if (trackBar21.Value == 0)
    valor21 = "\0";
if (trackBar21.Value == 1)
    valor21 = "2";
if (trackBar21.Value == 2)
    valor21 = "d";

string valor22 = "\0";
if (trackBar22.Value == 0)
    valor22 = "\0";
if (trackBar22.Value == 1)
    valor22 = "2";
if (trackBar22.Value == 2)
    valor22 = "d";

string valor23 = "\0";
if (trackBar23.Value == 0)
    valor23 = "\0";
if (trackBar23.Value == 1)
    valor23 = "2";
if (trackBar23.Value == 2)
    valor23 = "d";

string valor24 = "\0";
if (trackBar24.Value == 0)
    valor24 = "\0";
if (trackBar24.Value == 1)
    valor24 = "2";
if (trackBar24.Value == 2)
    valor24 = "d";

string valor25 = "\0";
if (trackBar25.Value == 0)
    valor25 = "\0";
if (trackBar25.Value == 1)
    valor25 = "2";
if (trackBar25.Value == 2)
    valor25 = "d";

string valor26 = "\0";
if (trackBar26.Value == 0)
    valor26 = "\0";
if (trackBar26.Value == 1)
    valor26 = "2";
if (trackBar26.Value == 2)
    valor26 = "d";
```

```
string valor27 = "\0";
if (trackBar27.Value == 0)
    valor27 = "\0";
if (trackBar27.Value == 1)
    valor27 = "2";
if (trackBar27.Value == 2)
    valor27 = "d";

string valor28 = "\0";
if (trackBar28.Value == 0)
    valor28 = "\0";
if (trackBar28.Value == 1)
    valor28 = "2";
if (trackBar28.Value == 2)
    valor28 = "d";

string valor29 = "\0";
if (trackBar29.Value == 0)
    valor29 = "\0";
if (trackBar29.Value == 1)
    valor29 = "2";
if (trackBar29.Value == 2)
    valor29 = "d";

string valor30 = "\0";
if (trackBar30.Value == 0)
    valor30 = "\0";
if (trackBar30.Value == 1)
    valor30 = "2";
if (trackBar30.Value == 2)
    valor30 = "d";

string valor31 = "\0";
if (trackBar31.Value == 0)
    valor31 = "\0";
if (trackBar31.Value == 1)
    valor31 = "2";
if (trackBar31.Value == 2)
    valor31 = "d";

string valor32 = "\0";
if (trackBar32.Value == 0)
    valor32 = "\0";
if (trackBar32.Value == 1)
    valor32 = "2";
if (trackBar32.Value == 2)
    valor32 = "d";
```

```
string valor33 = "\0";
if (trackBar33.Value == 0)
    valor33 = "\0";
if (trackBar33.Value == 1)
    valor33 = "2";
if (trackBar33.Value == 2)
    valor33 = "d";

string valor35 = "\0";
if (trackBar35.Value == 0)
    valor35 = "\0";
if (trackBar35.Value == 1)
    valor35 = "2";
if (trackBar35.Value == 2)
    valor35 = "d";

string valor36 = "\0";
if (trackBar36.Value == 0)
    valor36 = "\0";
if (trackBar36.Value == 1)
    valor36 = "2";
if (trackBar36.Value == 2)
    valor36 = "d";

string valor37 = "\0";
if (trackBar37.Value == 0)
    valor37 = "\0";
if (trackBar37.Value == 1)
    valor37 = "2";
if (trackBar37.Value == 2)
    valor37 = "d";

string valor38 = "\0";
if (trackBar38.Value == 0)
    valor38 = "\0";
if (trackBar38.Value == 1)
    valor38 = "2";
if (trackBar38.Value == 2)
    valor38 = "d";

string valor39 = "\0";
if (trackBar39.Value == 0)
    valor39 = "\0";
if (trackBar39.Value == 1)
    valor39 = "2";
if (trackBar39.Value == 2)
    valor39 = "d";
```

```
string valor40 = "\0";
if (trackBar40.Value == 0)
    valor40 = "\0";
if (trackBar40.Value == 1)
    valor40 = "2";
if (trackBar40.Value == 2)
    valor40 = "d";

string valor41 = "\0";
if (trackBar41.Value == 0)
    valor41 = "\0";
if (trackBar41.Value == 1)
    valor41 = "2";
if (trackBar41.Value == 2)
    valor41 = "d";

string valor42 = "\0";
if (trackBar42.Value == 0)
    valor42 = "\0";
if (trackBar42.Value == 1)
    valor42 = "2";
if (trackBar42.Value == 2)
    valor42 = "d";

string valor43 = "\0";
if (trackBar43.Value == 0)
    valor43 = "\0";
if (trackBar43.Value == 1)
    valor43 = "2";
if (trackBar43.Value == 2)
    valor43 = "d";

string valor44 = "\0";
if (trackBar44.Value == 0)
    valor44 = "\0";
if (trackBar44.Value == 1)
    valor44 = "2";
if (trackBar44.Value == 2)
    valor44 = "d";

string valor45 = "\0";
if (trackBar45.Value == 0)
    valor45 = "\0";
if (trackBar45.Value == 1)
    valor45 = "2";
if (trackBar45.Value == 2)
    valor45 = "d";
```

```
    string valor46 = "\0";
    if (trackBar46.Value == 0)
        valor46 = "\0";
    if (trackBar46.Value == 1)
        valor46 = "2";
    if (trackBar46.Value == 2)
        valor46 = "d";

    string valor47 = "\0";
    if (trackBar47.Value == 0)
        valor47 = "\0";
    if (trackBar47.Value == 1)
        valor47 = "2";
    if (trackBar47.Value == 2)
        valor47 = "d";

    string valor48 = "\0";
    if (trackBar48.Value == 0)
        valor48 = "\0";
    if (trackBar48.Value == 1)
        valor48 = "2";
    if (trackBar48.Value == 2)
        valor48 = "d";

    string valor49 = "\0";
    if (trackBar49.Value == 0)
        valor49 = "\0";
    if (trackBar49.Value == 1)
        valor49 = "2";
    if (trackBar49.Value == 2)
        valor49 = "d";

    string valor50 = "\0";
    if (trackBar50.Value == 0)
        valor50 = "\0";
    if (trackBar50.Value == 1)
        valor50 = "2";
    if (trackBar50.Value == 2)
        valor50 = "d";

    string final = string.Concat(valor1, valor2, valor3, valor4, valor5, valor6,
    valor7, valor8, valor9, valor10, valor11, valor12, valor13, valor14, valor15,
    valor16, valor18, valor19, valor20, valor21, valor22,bvalor23, valor24, valor25,
    valor26, valor27, valor28, valor29, valor30, valor31, valor32, valor33, valor35,
    valor36, valor37, valor38, valor39, valor40, valor41, valor42, valor43, valor44,
    valor45, valor46,valor47, valor48, valor49, valor50);

    serialPort1.Write(final);
}
```

```
        private void label51_Click(object sender, EventArgs e)
        {

        }

        private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
        {

        }

        private void label45_Click(object sender, EventArgs e)
        {

        }
    }
}
```

C Código del menú principal de la aplicación de Windows.

```
/*****************************************************************************************************
** ARCHIVO: home.cs                                                                                 **
**                                                                                                  **
** Archivo que contiene el codigo del menu.                                                         **
**                                                                                                  **
** AUTOR: Javier de Pedro Pascual                                                                   **
** FECHA: 08/02/2020                                                                                **
**                                                                                                  **
*****************************************************************************************************/
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Threading;
using System.IO.Ports;


namespace Prueba
{
    public partial class Home : Form
    {

        public Home()
        {
            //Creamos un hilo
            Thread t = new Thread(new ThreadStart(SplashStart));

            //Arrancamos el hilo
            t.Start();

            //Ponemos a dormir la forma principal
            Thread.Sleep(2000);

            //Finalizamos el hilo
            t.Abort();

            InitializeComponent();

        }

        public void SplashStart()
        {
            Application.Run(new splash());
        }
```

```
private void Home_Load(object sender, EventArgs e)
{

}

private void button1_Click(object sender, EventArgs e)
{
    Form1 miforma = new Form1();

    miforma.ShowDialog();
}

private void label1_Click(object sender, EventArgs e)
{

}


private void button7_Click(object sender, EventArgs e)
{
    Form1 miforma = new Form1();
    miforma.ShowDialog();
}

private void label2_Click(object sender, EventArgs e)
{

}

private void button2_Click_1(object sender, EventArgs e)
{
    Form2 miforma2 = new Form2();
    miforma2.ShowDialog();
}

private void button6_Click(object sender, EventArgs e)
{
    Form6 miforma6 = new Form6();
    miforma6.ShowDialog();
}

private void button3_Click(object sender, EventArgs e)
{
    Form3 miforma3 = new Form3();
    miforma3.ShowDialog();
}

private void button5_Click(object sender, EventArgs e)
{
    Form5 miforma5 = new Form5();
    miforma5.ShowDialog();
}
```

```
        private void button4_Click(object sender, EventArgs e)
        {
            Form4 miforma4 = new Form4();
            miforma4.ShowDialog();
        }

        private void button7_Click_1(object sender, EventArgs e)
        {
            Form7 miforma7 = new Form7();
            miforma7.ShowDialog();
        }

        private void button8_Click(object sender, EventArgs e)
        {
            Form8 miforma8 = new Form8();
            miforma8.ShowDialog();
        }
    }
}
```